

NAVAL POSTGRADUATE SCHOOL MONTEREY, CALIFORNIA



THESIS

**THE DOLPHIN DIDACTIC DATABASE SYSTEM
(DODDS)**

by

Michael Lyn Dodds

September 1997

Thesis Advisor:

Robert McGhee

Approved for public release; distribution is unlimited.

19980106 027

DTIC QUALITY INSPECTED 4

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE September 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis		
4. TITLE AND SUBTITLE THE DOLPHIN DIDACTIC DATABASE SYSTEM (DODDS) (U)		5. FUNDING NUMBERS		
6. AUTHOR(S) Dodds, Michael, L.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
<p>13. ABSTRACT (maximum 200 words)</p> <p>The Naval Command, Control and Ocean Surveillance Center (NCCOSC) Research, Development, Test and Evaluation Division (NRaD) Marine Mammal Research Programs are conducted by the Marine Mammal Research & Development Branch (D351). Progeny is a project, under D351, that trains Atlantic Bottlenose Dolphins (Tursiops Truncatus). Progeny was designed to explore the standardization of training, husbandry, and record keeping techniques that contribute to preparing, operating, and maintaining dolphins for Fleet systems.</p> <p>Presently, hand written forms are filled out to record data as trainers conduct their training exercises. These forms become the source for creating reports. The current data collection process is tedious, time-consuming, and potentially unmanageable for the staff.</p> <p>This thesis project provides a means to organize, gather, and maintain all the historical, current, and future data for the Progeny project. Capabilities are needed to gather the data so that timely, meaningful information, such as reports and graphs, can be made accessible to the staff.</p> <p>The deliverables from this study are the development of a relational database system for organizing and storing Progeny's data, and the development of an application for entering and accessing the data within the relational database. Output reports and graphs provide information from the data.</p>				
14. SUBJECT TERMS Dolphins Progeny DODDS.			15. NUMBER OF PAGES 132	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

Approved for public release; distribution is unlimited.

THE DOLPHIN DIDACTIC DATABASE SYSTEM (DODDS)

Michael Lyn Dodds
B.A., Point Loma Nazarene College, 1994

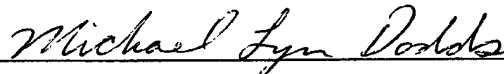
Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

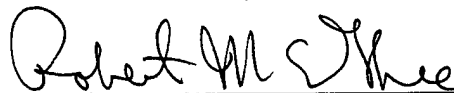
**NAVAL POSTGRADUATE SCHOOL
September 1997**

Author:



Michael Lyn Dodds

Approved by:



Robert McGhee, Thesis Advisor



Patrick Moore, Co-Advisor



Ted Lewis, Chair
Department of Computer Science

ABSTRACT

The Naval Command, Control and Ocean Surveillance Center (NCCOSC) Research, Development, Test and Evaluation Division (NRaD) Marine Mammal Research Programs are conducted by the Marine Mammal Research & Development Branch (D351). Progeny is a project, under D351, that trains Atlantic Bottlenose Dolphins (*Tursiops Truncatus*). Progeny was designed to explore the standardization of training, husbandry, and record keeping techniques that contribute to preparing, operating, and maintaining dolphins for Fleet systems.

Presently, hand written forms are filled out to record data as trainers conduct their training exercises. These forms become the source for creating reports. The current data collection process is tedious, time-consuming, and potentially unmanageable for the staff.

This thesis project provides a means to organize, gather, and maintain all the historical, current, and future data for the Progeny project. Capabilities are needed to gather the data so that timely, meaningful information, such as reports and graphs, can be made accessible to the staff.

The deliverables from this study are the development of a relational database system for organizing and storing Progeny's data, and the development of an application for entering and accessing the data within the relational database. Output reports and graphs provide information from the data.

TABLE OF CONTENTS

I.	INTRODUCTION	1
A.	PROGENY	1
B.	DATA COLLECTION METHODS.....	1
C.	SCOPE	2
II.	DEVELOPMENT STRATEGY.....	3
A.	DATABASE SELECTION	3
B.	LANGUAGE SELECTION	4
C.	TRADE-OFFS.....	5
D.	SUMMARY	6
III.	PROGRAM DESIGN	7
A.	DATABASE/TABLES/RELATIONSHIPS.....	7
1.	Interacting with the End Users	7
2.	Designing the Tables	7
3.	Designing the Relationships.....	13
B.	USER INTERFACE.....	15
1.	Interacting with the End Users	15
2.	Designing the Interface.....	17
C.	CHALLENGES AND SOLUTIONS.....	20
1.	Input Requirements	20
2.	Generating Reports.....	22
3.	Graphical Output	24
4.	Flexibility for Changes	25
5.	Database Integrity.....	25
D.	SUMMARY.....	26
IV.	PROGRAM DEVELOPMENT	29
A.	DATABASE/TABLE/RELATIONSHIP BUILDING	29
1.	Using Visual Basic's Data Manager.....	29
2.	Recordsets	30
3.	Foreign Keys, Primary Keys, and Indexes.....	31
B.	TABLE MAINTENANCE CAPABILITIES	31
1.	Building/Defining the Relational Tables.....	31
2.	Building/Defining the User-Defined Tables	34
3.	Database Integrity.....	35
4.	Record Searching Capabilities	36
C.	DATA COLLECTION SCREENS	36
1.	Two Primary Screens	36
2.	Saving the Data	38
D.	REPORT/GRAPH GENERATION	40
1.	Using Crystal Reports.....	40

2. Record Sorting.....	41
3. Using Visual Basic's Graph Utility.....	42
E. SUMMARY.....	43
V. PROGRAM OUTPUT.....	45
A. REPORTS AVAILABLE.....	45
B. GRAPH OUTPUT OPTIONS.....	47
C. OTHER OUTPUT ALTERNATIVES.....	49
D. SUMMARY.....	50
VI. CONCLUSIONS AND RECOMMENDATIONS.....	51
A. USER SATISFACTION.....	51
B. IMPROVEMENT POSSIBILITIES.....	52
1. Record Searching Techniques.....	53
2. Program Error Trapping.....	55
3. Input Screens.....	57
4. Single-User vs. Networked System.....	57
5. Recording the Actual Trial Data.....	58
APPENDIX. DODDS SOURCE CODE.....	59
LIST OF REFERENCES.....	117
INITIAL DISTRIBUTION LIST.....	119

LIST OF FIGURES

1. Progeny's Table Layout and the Defined Relationships.....	8
2. Main Menu.....	16
3. Forms/Databases/Tables.....	19
4. Output Reports/Graphs.....	20
5. Daily/Session Input Screen.....	21
6. Sample Output Report.....	23
7. Monthly Input Form.....	37
8. Typical Characteristic Input Form.....	40
9. Sample Output Graph.....	48
10. Session Records Form.....	52

ACKNOWLEDGMENT

The author would like to thank Professor Robert McGhee for his willingness to be the advisor for this thesis and for his overall guidance, careful reading, and help in seeing that the thesis is routed to the appropriate administrative personnel for approval prior to graduation.

The author would also like to thank the Marine Mammal Research and Development Branch for their willingness to provide thesis work for the Software Engineering effort at NRaD. The author would also like to extend special thanks to Patrick Moore, Randy Brill, Mark Xitco, Scott Klappenback, and Joy Ross for their support, guidance, patience, and helpfulness in making this thesis possible.

The author also gives his thanks to Sandy Ernst for her dedication and many hours of volunteer work spent editing the rough draft of this thesis.

I. INTRODUCTION

A. PROGENY

The Progeny project is designed to explore the standardization and potential enhancement of training, husbandry, and record-keeping techniques that could contribute to the overall efficiency of preparing, operating, and maintaining dolphins for Fleet systems. "Marine mammal systems (MMS) are an unusual, effective and unique solution to current problems of mine and obstacle hunting." [Ref. 1]

The Naval Command, Control and Ocean Surveillance Center RDT&E laboratory has been conducting experiments with dolphins since 1959. "The research and development has been directed towards two major goals: (1) training marine mammals as working animals to perform tasks for the direct accomplishment of Navy missions, and (2) building hardware based on the natural echolocation capabilities of these animals." [Ref. 1] The Progeny project takes advantage of the availability of naive, captive-born juvenile dolphins uncommitted to the associated time constraints on training schedules that are typically imposed on Fleet systems.

The Progeny project provides an opportunity to explore alternative behavioral and husbandry training techniques that could pay off in terms of improved training and maintenance efficiency and a reduction in manpower per animal requirements for Fleet systems. "The ability exists now to develop an MMS for support of amphibious operations that could be delivered from over the horizon and utilized for rapid reconnaissance of shallow and very shallow water areas to detect mined areas and to determine the relative densities of detected minefields." [Ref. 1]

B. DATA COLLECTION METHODS

Currently, Progeny has established, and utilizes, a uniform set of record-keeping tools to monitor dolphin care and training. Some of the current tools being used still require automating to enhance the effectiveness of the data being maintained. The current

tools can be reviewed and consolidated into a system to be adopted as a standard across the animal care and maintenance tasks conducted by Code D351.

A Progeny trainer's time with a dolphin, for training one particular behavior is known as a *Session*. Data collected from these sessions are recorded throughout the day by multiple trainers once a session is completed. Currently, Progeny session data is recorded manually into each dolphin's log book and onto log sheets.

Many of the sessions performed by the trainers are conducted in enclosures to allow easy access to the dolphins during the data collection process. "Pier-side work rooms with electrical hookups are available for set-up and storage of equipment and for conducting data collection, observations, etc." [Ref. 2]

C. SCOPE

The scope of this thesis is to develop an object-oriented, windows-based application that will automate the data collection procedures for the Progeny project, as well as automate the report writing and line graphs produced by the Progeny staff.

If information can be produced more quickly from the Progeny data available, this could lead to an increase in efficiency in preparing the Progeny dolphins for the Fleet systems.

The research of this thesis has resulted in two primary products. The first product is the written portion of the thesis explaining how the application was developed for the Progeny project and the design issues encountered along the way. The second product is the Visual Basic source code used to compile the application for the Progeny project (Appendix). The application program created for the Progeny project is named the Dolphin Didactic Database System (DODDS).

II. DEVELOPMENT STRATEGY

A. DATABASE SELECTION

In choosing a database suitable for the Progeny project, selection of a relational database that is 32-bit, Windows 95/NT compatible was desired to take advantage of the performance available with the 32-bit hardware. The Progeny project requires the database to have the ability to build and maintain multiple tables as well as display and build relationships among them. "A table is a logical grouping of related information arranged in rows and columns." [Ref. 3] The ability to create primary keys, foreign keys, and indexes was also a requirement of the database for handling issues such as uniqueness, fast retrieval, and sorting of the Progeny's records.

Another decisive factor in determining a database for this project was the ease and ability to retrieve the data once it is stored in the database tables. Visual Basic allows the creation of database tables that handle field and record selection by accepting query capabilities using Structured Query Language (SQL) statements. SQL statements are used for complex report generation, selecting data from multiple tables within the database.

SQL has evolved into the most widely accepted means to 'converse' with a database. Basically, the user asks questions in the SQL language; this is called a 'query.' The database engine 'answers' by returning any database rows that meet the requirements of the query. The query usually contains the names of the tables to search, the names of the columns to return, and other information that sets the scope of the search. [Ref. 3]

An additional selection factor in selecting the database was a user-friendly front-end enabling ease of use and maintainability by the developer. The database selected also had to be compatible with the software application language decided upon for creation of the actual program source code.

Microsoft's Visual Basic Jet Database Engine was chosen for this particular application because of the ease of designing and building a relational database that was 32-bit compliant and that would support 32-bit software features.

B. LANGUAGE SELECTION

Language selection has directed the way various organizations conduct their business. A lack of the required resources prevent an organization or a person from becoming fully competent in all programming languages available.

There are many ways to build applications. You can use a full-scale programming language such as COBOL, C, or C++. There are many such languages, each with its advantages and disadvantages. Programming with these languages gives you a great deal of flexibility and performance, but you work hard to use that flexibility or to get that performance.[Ref. 4]

In determining the language to use with the Progeny project, certain factors were taken into account. A user-friendly, object-oriented, Graphical User Interface (GUI) development language was preferred for rapid application development. The language selected also had to produce an end user GUI application for simplicity of use. A GUI can be defined as, "A graphics-based user interface that incorporates icons, pull-down menus and a mouse".[Ref. 5] The GUI is the interface between man and machine that allows man to interact with the application using a graphical interface. The ability for the language to produce an "executable" for ease of implementation was also considered in selecting an appropriate language.

The language selected should take advantage of the performance of the 32-bit operating systems for the development of fast running program executables. Also, the development language selected must be compatible with the relational database selected for the project. Since Microsoft's Visual Basic met all the above conditions, and includes Microsoft's Jet Database Engine, the best and logical choice was to choose Microsoft's Visual Basic programming language for the Windows application that needed to be developed for the Progeny project.

Visual Basic helps you to be more productive by providing appropriate tools for the different aspects of GUI development. You create the graphical user interface for your application by drawing objects in a graphical way. You set properties on these objects to refine their appearance and behavior. Then you make this interface react to the user by writing code that responds to events that occur in the interface.[Ref. 3]

C. TRADE-OFFS

Other databases and languages were also available that met the evolving requirements, developed in a series of meetings at NRaD, for the Progeny project. The major requirements included developing an application that collected the data provided by the Progeny project, producing report and graphical outputs so that information can be obtained from the data, and making the system easy to use for the trainers.

Some development applications may provide more language constructs for building the application. Others may offer more graphical output options or may provide better compilers for faster executing programs. It is rare that one development application will offer the best of everything once all things are considered, so a decision must be made as to the best application development language suited for the program under development.

Since Visual Basic met the requirements of the users, provides all the utilities necessary for development, has the capability to compile fast executing programs and has simple language constructs, Visual Basic was the application language chosen for development of the DODDS application.

Once the language and database issues had been decided upon, another question had to be resolved. That question was what development strategy was to be used during the development phases of this application. Two of the most common strategies to use during development are the *Waterfall Life Cycle* and the *Spiral Life Cycle*.

In the waterfall strategy, "The fundamental idea is to divide the development process into a series of phases or stages, each of which finishes before the next one starts".[Ref. 4] The purpose of the waterfall model structures the development process into an ordered series of goal-directed processes. "The principles behind the model remain: to perform certain processes and to minimize costs by doing them in a specific order".[Ref. 4]

The spiral life cycle is a variant of the waterfall model that adds prototyping and risk management to the waterfall processes. It assumes that you still need all the waterfall processes, but that doing them singly, in order, is not required.[Ref. 4]

The spiral starts out with a risk analysis and prototype, which leads to a concept document. Another risk analysis and prototype precedes full requirements, and another precedes a high-level design. The final iteration follows a final risk analysis and operational prototype with detailed design, coding, and the other parts of the waterfall life cycle, much reduced in scope. [Ref. 4]

The strategy that was chosen for this development process was the spiral method. This development process required a lot of interaction between the developer and the user because of the unfamiliarity of each other's domain. Many prototypes were built prior to the finished product in order to help all involved begin to visualize the requirements that have been implemented and those that were still required.

D. SUMMARY

Much thought is required when selecting the correct database format and determining the appropriate language to use when developing a new computer application. The most important issue is to make sure the application can be designed to meet the user's needs. Additional things to take into consideration are ease of use, the hardware that the application will be installed on, ease of development, and ease of maintenance when future enhancements are needed.

Once a database format and a programming language have been selected for developing the application, program design is the next phase in developing the application. Design issues include, but not limited to, such things as: interviews with the users to discuss their needs, obtaining knowledge of the current way business is conducted, designing the application tables and relationships, and designing a interface that is intuitive to use and is amiable to the user.

III. PROGRAM DESIGN

A. DATABASE/TABLES/RELATIONSHIPS

1. Interacting with the End Users

The interaction between the end user and the developer is very important if an application is to be developed that meets the user's needs. "A large amount of interaction with the customer and the users is needed to formalize the requirements, because the developers are usually not experts on the customer's problem and may not be familiar with the special terminology of the area, or may have informal interpretations for some of the terms that differ from the customer's intended meaning".[Ref. 6]

In designing the database tables and their interactions, many meetings were conducted with the end users to assure correctness in how the data variables are associated with each table defined, and the way the tables are related to one another. Some training of the users was necessary to explain the concepts of a relational database, tables, relationships, and elements. The meetings were thought provoking, with the payoff being that, users will receive a product that meets their needs.

After the initial meeting with the end users to discuss the application domain and to look at the data collection forms, a preliminary overall database design was developed. The design showed what was interpreted as the end users way of doing business and became the basis for the creation of the database tables and ultimately the DODDS application.

2. Designing the Tables

The final design of the DODDS application includes ten related tables and three non-related, user-defined tables (Figure 1). The ten related tables follow the typical guidelines of developing a relational database system. "Basically, a relational database is one that stores data of tables, made up of columns and rows of data."[Ref. 3] The three

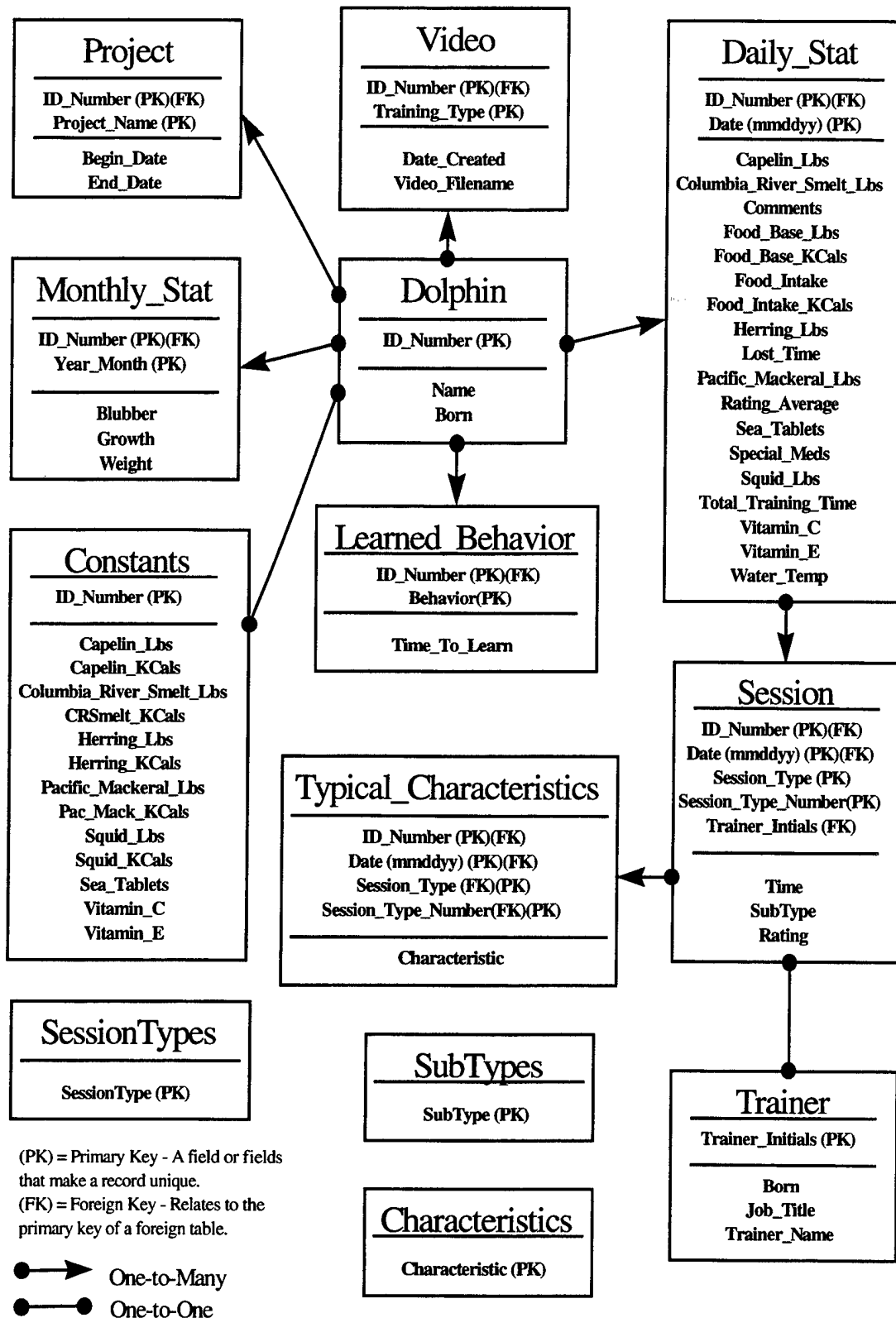


Figure 1. Progeny's Table Layout and the Defined Relationships.

user-defined tables have no relationships to any other tables. The sole purpose of the three user-defined tables is for building values within particular fields. These fields will ultimately be used to populate lookup tables.

Within the Progeny database, the ten related tables are: Project, Monthly_Stat, Constants, Video, Dolphin, Learned_Behavior, Daily_Stat, Session, Trainer, and Typical_Characteristics. The three user-defined tables are: Session_Types, SubTypes, and Characteristics. Each is discussed separately below.

Dolphin is the main table, and has one-to-many and one-to-one relationships to other tables associated with it. The Dolphin table maintains the records of each dolphin. The Dolphin table's field names are ID-Number, Name, and Born. "Each column in a database table is called a field." [Ref. 3] The first two digits of the ID-Number represents the species. In this case, the first two digits are "Tt", which stands for Tursiops Truncatus. The next three digits of the ID-Number represents a serial number with no special meaning. The last digit of the ID-Number tells the gender of the dolphin: "M" for male, "F" for female. Name is the name of the dolphin. Born is the birth date.

The Project table holds the names of the projects of which a dolphin has been a part. The Project table contains Project-Name, ID-Number, Begin-Date, and End-Date. The ID-Number field is explained under the Dolphin table description above. The Project-Name tells the trainers the other projects with which the dolphin has been involved. Knowing the other projects informs the trainers about training a particular dolphin that has already accomplished, and the other animals with which the dolphin has been in contact. This type of data is good for determining under what circumstances and company a dolphin performs best.

The Begin-Date and End-Date fields associated with the project records provides valuable information to the Progeny staff. This information includes the number and types of concurrent or non-concurrent training projects a dolphin has been on, the sequence of training projects to which a dolphin has been subjected, and the length of time spent on each project.

The table named Monthly_Stat is short for Monthly Statistics. Each month a dolphin is measured in three different ways: blubber, growth, and weight. The

Monthly_Stat table contains the ID-Number, Year-Month, Blubber, Growth, and Weight fields. The ID-Number field is explained under the Dolphin table description. The Year-Month field is a four digit field which has been defined as YYMM for record sorting purposes. Since YY is only a two-digit field, after the year 2000, records will show up at the beginning of ascending-order sorted reports. Blubber is a field that contains a measure of the blubber thickness a particular dolphin has at a particular measuring point. It is measured in centimeters. The Growth field is the length of the dolphin, also measured in centimeters, and the weight is how much the dolphin weighs, measured in pounds.

Video is a table that is defined to keep track of the videos taken of a particular dolphin during the execution of trained behaviors. This table consists of an Id-Number, Training-Type, Date-Created, and Video-Filename. The ID-Number field is explained under the Dolphin table description. The Training-Type is a particular type of training that a dolphin has been trained in and performs successfully. The Training-Type is also known as a *behavior*. The Date-Created is the date when the video was taken of the dolphin performing the particular behavior. The Video-Filename is a name given to the video for tracking, storing, and retrieving purposes.

The Learned_Behavior table maintains the behaviors that a dolphin has proven to have mastered. The Learned_Behavior table consists of an ID-Number, Behavior, and Time-to-Learn. The ID-Number field is explained under the Dolphin table description. The Behavior field indicates the type of behavior(s) that the dolphin has been taught and is competent in. Behaviors fall into one of three main categories: Basic, Husbandry, and Systems behaviors. Under each of these three main categories are approximately eight specific types of behaviors. An example of some of the behaviors types include: Stationing, Gating, Mouth/Eye Inspection, Blood Sampling, Beaching, Marker Delivery, and Boat Follow.

Another table is named Trainer, which identifies the trainers as the ones who conducted the sessions at NRaD. It consists of Trainer-Initials, Born, Job-Title, and Trainer-Name. Trainer-Initials is the initials of the trainer's name in First, Middle, Last name format. Born is the birth date of the trainer and is used to compute the age of the

trainer. Born could also be used for statistical purposes such as the average age of a dolphin trainer. So far, the age of the trainer has not been used as a factor when computing statistics. Job-Title simply describes the dolphin trainer's position within the hierarchical structure of the department. Trainer-Name is the full name of the trainer in First, Middle, Last name format.

Typical_Characteristics is the name of a table that holds all the events or behaviors the dolphin displayed during the session. The Typical_Characteristics table consists of an ID-Number, Date, Session-Type, Session-Type-Number, and Characteristic. The ID-Number field is explained under the Dolphin table description. The Date field is the day the session took place. The Session-Type reflects the type of session taking place. Session Types are defined by the trainers and currently fall under four main categories: Training, Play, Enrichment, and Exercise. The Session-Type-Number field is used to distinguish between multiple session types of the same type conducted within the same day. The Characteristic field is a particular characteristic demonstrated by the dolphin during a session. Multiple characteristics can be displayed during a session.

The Constants table is used to store data that is relatively constant from day to day. The information contained within this table is used to populate the daily statistics record on user request. The trainer's time and data entry requirements are reduced significantly by loading these constants into their daily statistical records. Most of the fields pertain to each dolphin's fish intake and kilo-calories consumed per day. The Constants table contains the following fields: ID-Number, Capelin-Lbs, Capelin-Kcals, Columbia-River-Smelt-Lbs, CRSmelt-Kcals, Herring-Lbs, Herring-Kcals, Pacific-Mackeral-Lbs, Pac-Mack-Kcals, Squid-Lbs, Squid-Kcals, Sea-Tablets, Vitamin-C, and Vitamin-E.

The ID-Number field is explained under the Dolphin table description. Capelin-Lbs, Columbia-River-Smelt-Lbs, Herring-Lbs, Pacific-Mackeral-Lbs, and Squid-Lbs are the amount of fish, in pounds, allotted to a particular dolphin per day. Capelin-Kcals, CRSmelt-Kcals, Herring-Kcals, Pacific-Mackeral-Kcals, and Squid-Kcals is the amount of Kilo-Calories associated with the amount of the particular fish given. The Sea-Tablets,

Vitamin-C, and Vitamin-E fields represent the number of that type of pill given on a daily basis for a particular dolphin.

The Daily_Stat table is short for Daily Statistics. It contains data that is kept each day per dolphin. It holds the data that is entered only once per day. The fields of this table include: ID-Number, Date, Capelin-Lbs, Columbia-River-Smelt-Lbs, Herring-Lbs, Pacific-Mackeral-Lbs, Squid-Lbs, Food-Base-Lbs, Food-Base-Kcals, Food-Intake, Food-Intake-Kcals, Lost-Time, Rating-Average, Special-Meds, Total-Training-Time, Water-Temp, Comments, Sea-Tablets, Vitamin-C, and Vitamin-E.

The ID-Number field is explained under the Dolphin table description. The Date field represents the date to which the daily statistics information pertains. It has the following format: mmddyy. Capelin-Lbs, Columbia-River-Smelt-Lbs, Herring-Lbs, Pacific-Mackeral-Lbs, and Squid-Lbs are the amount of fish, in pounds, allotted to a particular dolphin for that particular day. Food-Base-Lbs and Food-Base-Kcals are the amount of fish in pounds and kilo-calories, respectively, that the dolphin is supposed to receive for that day (i.e., it is the amount of fish loaded into the dolphins food buckets in the morning). It does not necessarily mean that this is the amount the dolphin will receive before the day is through. The dolphin may refuse to eat his/her allotted portion or perhaps extra food may be given for some reason. On the other hand, Food-Intake and Food-Intake-Kcals do represent the actual total weight of the fish food eaten by the dolphin in pounds and kilo-calories, respectively. The difference in these fields may present useful information to the Progeny trainers and staff.

The Lost-Time field is the time the trainer feels was unproductive during the day due to such things as, dolphins not cooperating, or performing administrative duties that interfered with the trainer's normal schedule. Rating-Average is a field that is the average of all the sessions performed throughout the day. Each session throughout the day is rated on a scale of 1 to 5, and at the end of the day, all the ratings are added up and then divided by the number of sessions in that day. The result is then entered into the Rating-Average field. Special-Meds is a boolean field that simply states whether a dolphin is on special medication that particular day. Total-Training-Time is the total time of all sessions spent with a particular dolphin during the day. It does not include Lost-

Time. Water-Temp is short for water temperature. It is used for entering the ocean water's temperature for the day. Comments is a free-formatted text field for entering special comments that the trainer feels is important to record for that day.

3. Designing the Relationships

Non-relational database applications store their data in one or more databases. However, there is no method for grouping elements and there are no relationships setup between the databases. The method for developing a relational database is: designing the table structures, grouping the related elements together, and setting up the relationships between the tables.

Relational databases were a new way of thinking about how data should be structured and stored. The key to this type of database is in understanding the relationships within data, then structuring the information base to reflect those relationships.[Ref. 7]

"Instead of modeling the relationships in the data according to the way that it is physically stored, the structure is defined by establishing relations between simple tables."[Ref. 8] The DODDS application has ten tables that interact with one another through relationships (Figure 1).

The DODDS program contains nine relationships between the ten tables. The table named Dolphin contains six of these relationships with the other tables. Associated with each table are Primary Keys, Foreign Keys, and Indexes.

Tables are linked by what is called a Primary Key. A primary key is made up of one field or a set of fields within a table. "All values in the primary key must be unique, and there can be only one primary key for a table."[Ref. 8] Primary keys are used so that each record can be uniquely identified in a table, as well as for maintaining database integrity.

When a table needs to create an association with another table to access that other table's data, each of the tables involved must have the same field within it so that the calling table can locate the record(s) it requires. Since the concept of a relational database is to prevent redundant data, it is likely that the other table will not have a field which is a

duplicate of the calling table's primary key. To resolve this problem, a new field is placed in the called table that matches the primary key of the calling table. Although this produces redundancy of the data within the keys, it is the only way to create an association of records between tables. "The key that establishes the relation is called a foreign key, because it relates to the primary key of a 'foreign' table." [Ref. 8]

Indexes are used for searching and sorting so that records can be retrieved more quickly and arranged in some predetermined order. "Each index entry points back to the database row it references." [Ref. 3] Indexed records also appear to be sorted when accessed or displayed. A table may contain none or many indexes. An index is based on a field or combination of fields within a table. Indexes are created at design time, but are selected at run time during the execution of a program. The primary key can also be used as the index of the table.

Relationships are either one-to-one or one-to-many. A one-to-one relationship means that for each record in the calling table, there is only one record in the called table that satisfies the search request. If more than one record is located by the calling program in a one-to-one relationship between two tables, a database integrity error has occurred and appropriate corrective action should be taken. A one-to-many relationship means that for each record in the calling table, there can be zero or more records in the called table that satisfy the search request.

The Progeny table design contains nine relationships between the ten tables. The Dolphin table acts as the calling table in six of these relationships. The Dolphin table has a single one-to-one relationship and five one-to-many relationships associated with it. The one-to-one relationship is to the table named Constants. The relationship can be read as, "for each dolphin record, there is exactly one constant record associated with it."

The one-to-many relationships described above are to the tables named Video, Project, Monthly_Stat, Learned_Behavior, and Daily_Stat. The relationships to these tables can be read as, "for each dolphin record, there are many video records, project records, monthly statistical records, learned behavior records and daily statistical records that can be associated with it", respectively.

The Daily_Stat table has a one-to-many relationship with the table named Session. It can be read as, "for every daily statistic record, there may be many session records associated with it."

The Session table itself has two other relationships. One is with the Typical_Characteristics table and the other with the Trainer table. The relationship to the Typical_Characteristic table is a one-to-many relationship and can be read as, "for each session, there can be multiple typical characteristics associated with it." The relationship to the Trainer table is a one-to-one relationship and can be read as, "for each session, there is only one trainer associated with it."

B. USER INTERFACE

"The Visual Basic programming system allows you to create robust and useful applications that make use of the Graphical User Interface (GUI)."[Ref. 3] The DODDS program is completely developed using Visual Basic's GUI capabilities. The program has three main areas that are accessible from the main menu (Figure 2). The first area allows the user to enter his/her daily and monthly input records and provides access to all the tables in the program. The second area allows the user to generate reports and graphical outputs from the program. The third area is for accessing the video recordings of the dolphin behaviors. The video recordings portion of this program was not implemented due to lack of availability of dolphin behavior recordings. Selecting the video recording option will bring up an empty form that can be used at a later date for displaying the videos available.

1. Interacting with the End Users

When designing a system of which the analyst/programmer has no current knowledge, extensive time is spent in meetings with the end users in order to understand

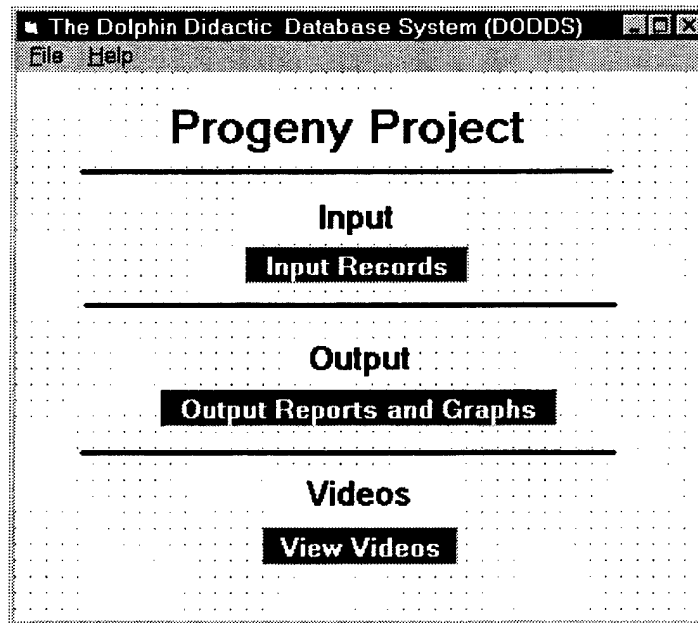


Figure 2. Main Menu.

the users' environment and discuss the expectations of the program about to be designed. Time is also spent in trying to understand the terminology of the application domain so that relationships and elements can be properly defined. During the design period, time is needed for reviewing and discussing the data collection forms currently used by the trainers and what all the information on the forms means.

Sometimes, procedures and recording techniques have to be standardized so that the new system can operate properly. Terminology also has to be well-defined so that input prompts and messages display the appropriate meanings.

The DODDS application was designed interactively with the users so that validation checks could be made at each stage of the design as it was being developed. This allowed the end users to develop their own program based on their specific needs. It also made the end users partially responsible for the design effort and ultimate outcome of the program.

2. Designing the Interface

Two tools were used for designing the interface. One tool involved taking advantage of previous programming knowledge. The other tool was to become familiar with the application domain and to obtain the data collection forms from the users so that the interface resembles the format they are currently using. If the user recognizes and feels comfortable with the program layout, he/she is more likely to adopt it into their environment and utilize it to the full potential.

At the conclusion of the design meetings, there were three primary data groupings. The data is either considered constant data or rarely changing, or the data is collected on a daily basis during the daily sessions, or the data is collected on a monthly basis during the dolphins monthly checkups. The constant data is written once and copied onto the daily forms on request. Daily data is collected on daily input forms. A monthly input form is used to collect the monthly data. When designing the interface, all three of these data collection techniques were taken into consideration.

The solution for the constant data was not difficult. A table was constructed that stores, modifies, and retrieves data used for inputting into the daily forms on request. Data is manipulated directly through pulling up the record in the table and making any necessary changes. New records are easily added and old records are easily deleted. Getting the data into the input forms is accomplished by a button on the input form to load the constant data for a particular dolphin, if desired.

The solution for the monthly and daily data collection was to create two separate input forms which are easily obtainable from the main menu since these options will probably be utilized a majority of the time by the trainers. These data collection forms comprise the first of three categories available from the Input forms menu.

Three main challenges had to be faced in designing the program input. First, historical data collected prior to this program will need to be easily entered into the tables defined. Entering the historical data will probably be the task of a secretary or a student-aide, either of which may be unfamiliar with the data collection techniques. Secondly, daily input can be entered into the defined tables by recording their daily input manually onto their current paper forms and then enter the information all at once at the end of their

sessions. The third way for trainers to input their data is for the trainers to enter it as it is collected; i.e., after each session with a dolphin. This is the most likely way the data will be entered into the database. This presents a potential problem of a trainer inadvertently saving another trainer's incomplete data currently displayed on the input screen, thereby creating duplicate records in the database. In addition, a trainer could inadvertently erase another trainer's data before the previous trainer had a chance to save the data into the tables.

Daily records for each dolphin are created only once each day, but multiple sessions per dolphin will be recorded throughout the day by several trainers working with more than one dolphin. Taking this into consideration meant that the interface needed to keep the daily records separate from the session records being input.

To assist the trainers from inadvertently entering incorrect or duplicate data, the program prompts the users when possible error situations arise. The input screen was also designed to look and behave like two separate input forms so that the daily records input area appears separated from the session records input area. Warning messages and error messages are displayed, when appropriate, to help the user make the correct decisions while saving data or exiting the input screens.

The main menu is divided into three primary parts. The first part, or the input part, handles the data collection forms and all input into the tables. The second part handles all output generated from the DODDS program, and the third part is where any videos taken of the dolphins can be selected and viewed. The input portion of the program is also subdivided into three sections (Figure 3). The above-mentioned input screens comprise the first of three parts from the Input forms menu. The second part of the Input forms menu gives the trainers the ability to add, change, or delete records directly into any of the remaining relational tables. The third part of the of the Input forms menu gives the trainers the ability to add, change, or delete records directly into any of the three user-defined, non-relational tables.

Access to the tables allows the users to double-check entries that were previously saved or to clean up incorrect records that were saved unintentionally. Direct

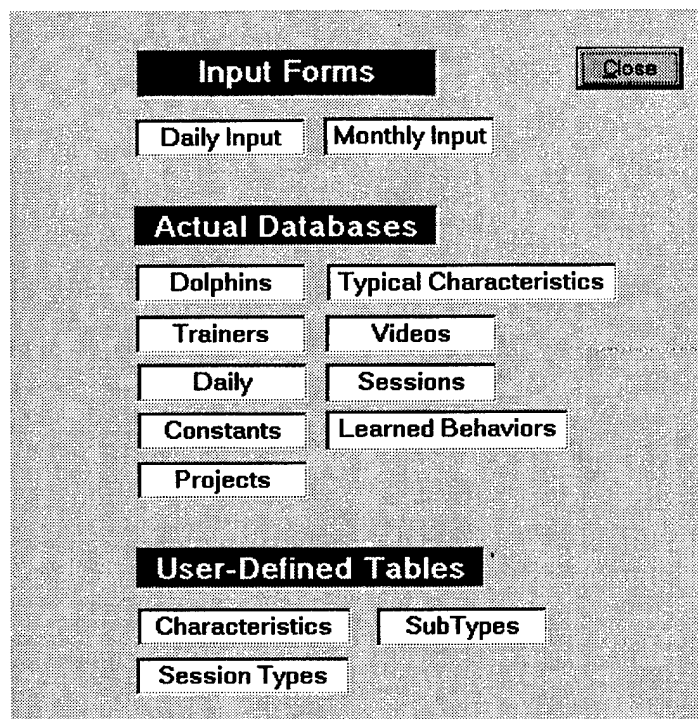


Figure 3. Forms/Databases/Tables.

links from the daily input screen to the pertinent tables were incorporated as well for convenience.

The output portion of the DODDS program is the second of three main topic areas accessible from the Main menu (Figure 4). From there, hard-copy reports are produced from each of the 13 tables. Pre-defined graphs as well can also be selected and displayed from this area that show relationships between selected variables.

The Video portion of the DODDS program is the last of three main topic areas accessible from the Main menu. So far, this area has not been addressed in the program portion of this thesis due to lack of availability of videos taken. A separate blank form was designed into the program to allow for future expansion.

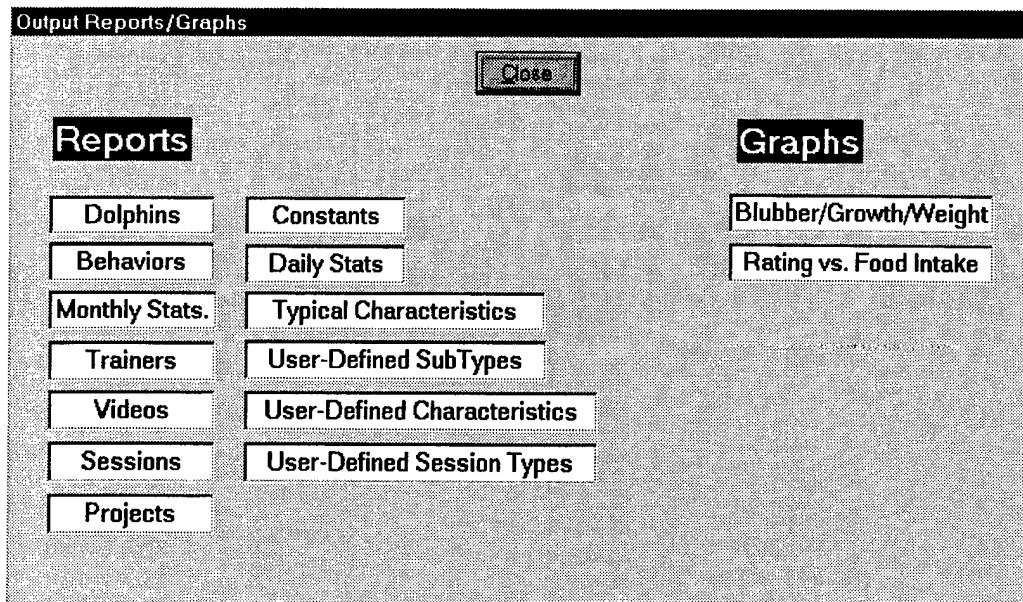


Figure 4. Output Reports/Graphs.

C. CHALLENGES AND SOLUTIONS

1. Input Requirements

The biggest input requirement challenge was designing one input form to be flexible enough to provide for both batch and incremental type inputs. Batch type inputs allow for all the day's data for each dolphin to be entered at the end of each day. Incremental type inputs allow for data to be input throughout the day by multiple trainers working with multiple dolphins. The trick in the incremental type input is to be sure that the information on the input screen is obvious enough so that other trainers will not become confused as to whose data is appearing on the screen (Figure 5).

The solution for batch input was to create the input screen with enough session entries to be able to handle twenty sessions per dolphin. This will allow ample room to input the day's session data. The daily input portion of the program was designed using two full screen pages for inputting the daily and session data for a dolphin (Figure 5).

The solution for the incremental type of input was more complicated. First the input form has to have the appearance of having a separate area for the daily data and the

Daily Input

ID_Number:

Load Default Values

Clear Form

Daily Input

Save

Exit

Date:

Herring_Lbs:

Pac_Mackerel_Lbs:

Capelin_Lbs:

CRiver_Smelt_Lbs:

Squid_Lbs:

Food_Intake:

Food_Intake_KCals:

Food_Base_Lbs:

Food_Base_KCals:

Special_Meds: ☐

Sea_Tablets:

Vitamin_E:

Vitamin_C:

Lost_Time: (Min.)

Total_Training_Time: (Min.)

Water_Temperature:

Rating_Average:

Comments:

View Daily Statistics

View Constants

Session Input

ID_Number:

Date:

Today

Session Type:

Session Number:

Sub Type:

1

2

3

4

5

Time:

Rating:

Trainer Initials:

Characteristics: ☐

Display Records

View sessions

More Input

View Typical Characteristics

Figure 5. Daily/Session Input Screen.

21

session data. This allows for working with two separate dolphins at the same time. The daily data can refer to one dolphin's data while the session data can refer to another dolphin's data. That way, when a different trainer sits down with the program, the trainer will not have to worry about dealing with the daily data of another trainer's dolphin if he/she is entering only session data. Or, if a trainer is dealing with only daily data for a particular dolphin, the trainer will not have to worry about another trainer's session data on the lower portion of the input form.

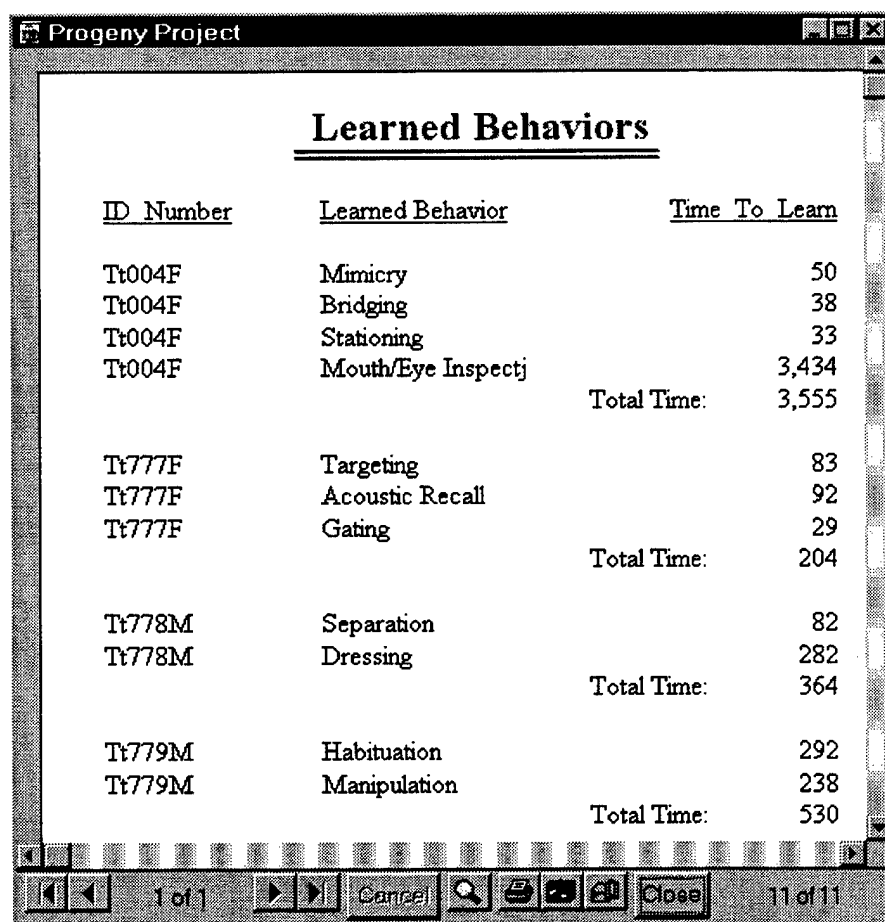
The separation of the input screens is accomplished by physical manipulation of the layout of the screen design. Prompts are also displayed to the user when different types of data are about to be saved. The input screens are designed so that the daily input portion is at the top of the form, with a solid line separating the top half from the bottom half. The bottom half of the input screen begins the input portion for the session data. The second input page is a continuation of the session input from the first page. The prompts that are displayed while the data is being saved give the trainer the choice of saving the daily data, session data, or both. Error messages are displayed to the user in order to help prevent input errors when duplicate records are detected. Easy access to all pertinent tables is also provided from the first page of the daily input screens so that table data can be viewed and manipulated.

2. Generating Reports

The challenge involved in generating reports dealt with accessing the data and determining what information needs to appear on the reports. Accessing the data proved to be a challenge because the data may reside in multiple tables. A simple listing of a table's records may not suffice in all cases.

Visual Basic has a report generating tool called Crystal Reports. This tool simplified the report writing significantly when it came to selecting, sorting and grouping records for report output. "Crystal Reports was created to allow technical and non-technical users to create customized reports quickly and easily, from a variety of databases." [Ref. 9]

In determining what data would appear on which reports required detailed discussion with the user since only they know what information is important to the overall goals of the Progeny project. The DODDS program creates 13 reports (Figure 6). The reports are a listing of each table, related and user-defined, sorted by each table's primary keys or a particular predefined index. When appropriate, the records are grouped based on changes in a particular field's value.



<u>ID Number</u>	<u>Learned Behavior</u>	<u>Time To Learn</u>
Tt004F	Mimicry	50
Tt004F	Bridging	38
Tt004F	Stationing	33
Tt004F	Mouth/Eye Inspectj	3,434
Total Time:		3,555
Tt777F	Targeting	83
Tt777F	Acoustic Recall	92
Tt777F	Gating	29
Total Time:		204
Tt778M	Separation	82
Tt778M	Dressing	282
Total Time:		364
Tt779M	Habituation	292
Tt779M	Manipulation	238
Total Time:		530

1 of 1 11 of 11

Figure 6. Sample Output Report.

Reports are management tools. Their purpose is to help individuals quickly grasp the essential elements and relationships found in raw data so they can make effective decisions. For a report to be effective, it has to present the right data in a logical way. If it presents the wrong data or if it presents the right data in a haphazard manner, the report may slow the decision making process or even encourage incorrect decisions.[Ref. 9]

3. Graphical Output

Another challenge is that graphical output is now being generated manually and there is a need to have the graphical output generated via the DODDS program itself. A way is needed to export data from the Progeny tables and use it for graphical output using different types of data that have different measuring scales. Line graphs are being generated manually that have multiple y-axes so that data points can be plotted next to one another showing relationships and variations between one variable and another.

The solution for producing a graphical output was found within Visual Basic. Visual Basic contains custom controls for producing graphics. "The graph control allows you to design graphs interactively on your forms." [Ref. 10] It is possible to send new data to the graphs at run time, print the graphs, or even copy the graphs onto the Clipboard for pasting into other applications. One can also change the graphs styles and shapes at run time.

One limitation of the graphic custom control within Visual Basic is the inability of Visual Basic to generate multiple y-axes for plotting points with different measuring scales on a single y-axis. For example, if a comparison is needed between a dolphin's total weight measured in pounds and the month's growth of the dolphin measured in centimeters, the points cannot be plotted unless a common y-axis scale is used for both sets of data points. The problem continues with weight ranging around 900 pounds, and growth, in most cases, ranging less than 20 centimeters. Plotting the data points will need a y-axis scale ranging from 0 to 1000. This is not practical when trying to see patterns between the two variables from a graphics standpoint. One solution could be eliminating the values on the y-axis and adding or subtracting a constant factor to one of the sets of data points. This technique would give the appearance of the data sets being closer to one another on the line graph and relational patterns could be spotted more easily.

However, as more graph types were made available, such as the Area Graphs and the Stacked Bar Graphs, the relationships between the data sets became more apparent. Therefore, no changes to the data sets were necessary.

4. Flexibility for Changes

In order for the DODDS program to be able to change with the Progeny project, and remain a usable application, flexibility had to be built into the program. Dolphins may be added or removed from the project, and trainers may come and go. New dolphin behaviors may also be observed and wish to be recorded. New session types may be required by the project and new training procedures added. A dynamic project mandates a dynamic application. Also, the application needs to be easily updated by the end user.

To solve this problem, the DODDS was designed to allow the user access to all the tables so that new records can be added and older ones deleted as necessary. Some of the tables are used to populate drop-down menus within the application. Adding new records to the appropriate tables will automatically update the drop-down menus. Some of the tables such as the Daily, Session, Constants, and Typical_Characteristic tables, are also accessible directly from the "daily input screen" to allow the user to view and make modifications as needed (Figure 5). This ability to change the databases gives the application flexibility it requires for it to be a useful tool for the Progeny project for a long time.

5. Database Integrity

Database integrity is a major concern with developers that design applications that record, store, and utilize data. Two possibilities of handling database integrity were available while designing the DODDS program. One possibility was to use Microsoft's built-in Referential Integrity capabilities provided by the Data Manager. The other possibility was to control the integrity of the database through programmer code.

Note that if you are using a Microsoft Access database or a native Jet database...Visual Basic will validate the accuracy of 'foreign' keys according to the restrictions set by the Relation object. Otherwise, you are responsible for verifying referential integrity yourself.[Ref. 3]

Utilizing Microsoft's built-in Referential Integrity capabilities proved to be very difficult while designing the DODDS application. Strict rules do not allow for flexibility

while design changes are being made during program development. Tables are developed in too rigid an environment for creativity and experimentation on the developer's part. Since dealing with the referential integrity capabilities proved to be more frustrating than helpful, the DODDS program does not utilize the referential integrity capabilities available. The solution was to add preventive measures to any object capable of updating the data contained in the tables. Program coding, used exclusively for data integrity, was included in all defined buttons capable of adding, changing or deleting records on any of the tables within the DODDS application. The button used for saving daily and session data also contains coding to maintain database integrity as records are added individually or in quantity into the tables.

D. SUMMARY

When designing an application, many things need to be taken into consideration. The first step requires getting acquainted with the end users to find out their programming needs. Familiarizing oneself with the user's application domain and the language used within helps to ensure the design of an understandable application.

Designing the database tables and the attributes contained within the tables is accomplished in the design phase. Designing the keys within a table and the table's relationships to one another needs to be established. Data integrity becomes a factor when designing the relationships between the tables. The design phase also needs to consider the interface between the user and the program. A GUI interface should be implemented when possible.

Throughout the development phase, many unexpected challenges will become apparent. A good design will be flexible and allow for modifications to be easily made. The expected output of the program needs to be addressed in the design phase so that useful information will be the outcome of the program. Output usually takes the form of a report or a graph representing the data within the tables.

The next chapter discusses the actual program development using Microsoft's Data Manager. Topics include the physical building of the database, the tables, and each

table's keys. Saving and sorting the data, record searching techniques, and database integrity issues are also mentioned in the following chapter. The programs data collection screens, and the output reports and graphs generated, are also discussed.

IV. PROGRAM DEVELOPMENT

A. DATABASE/TABLE/RELATIONSHIP BUILDING

1. Using Visual Basic's Data Manager

A Visual Basic database can be developed by simply running the Data Manager application that comes with Visual Basic. With the Data Manager, a Jet database can be created without programming. Once the database is created, it can be modified and manipulated. "The procedure for creating a new Jet database is simply a process of creating and defining data access objects that correspond to the tables, fields, indexes, and relations of your database design." [Ref. 8]

Once the Data Manager is executed, the process of building the tables begins. The developer is prompted for the table name, field names, field types, and field sizes. Field types that are available are: Boolean, Byte, Integer, Long Integer, Currency, Single, Double, Date/Time, Long Binary, Text, and Memo. Once the table is constructed, the Data Manager allows for indexes to be created. The developer also has the choice of entering records into the table once the table has been defined. The tool used to access the relational tables from within a Visual Basic program is called the "data control".

The data control provides a relational interface to database files. Basically, a relational database is one that stores data of tables, made up of columns and rows of data. In Visual Basic, columns are referred to as fields, and rows are referred to as records. [Ref. 3]

It is also from the Data Manager that Structured Query Language (SQL) statements can be built. SQL statements are used to access multiple tables and data fields. SQL statements are usually used for creating reports or displaying data from multiple tables. SQL statements can also be used within the program to create additional tables as needed at runtime.

2. Recordsets

When developing the DODDS program for the Progeny project, certain considerations had to be made as to what type of recordsets needed to be developed under what circumstances. "A Recordset object, as the name implies, is a set of records drawn from the database." [Ref. 8] A recordset can consist of all the records in a database table, a subset of all the records, or the records retrieved from an SQL query on the database table. There are three types of recordsets: Dynaset, Table, and Snapshot.

"A dynaset-type Recordset enables you to filter, sort, extract, and update data from more than one table." [Ref. 8] However, when querying data, filtering and sorting are best performed on smaller datasets. "It may be far more efficient to re-execute the query with a modified ORDER BY or WHERE clause." [Ref. 11] With a dynaset-type Recordset, data can be manipulated in the underlying tables. "A dynaset-type Recordset is a dynamic set of records that can contain fields from one or more tables or queries in a database and may be updatable." [Ref. 11] One disadvantage of the dynaset-type Recordset is that there is no guarantee the data retrieved will follow any specific order or sequence. "If you need to order your data, use an SQL statement with an ORDER BY clause to create the Recordset." [Ref. 11]

A table-type Recordset refers to a local table only. A reference to an attached table cannot be created. Indexing and the Seek method can be used on this type of recordset.

A snapshot-type Recordset contains a fixed copy of the data as it exists when the snapshot is created.

A snapshot can't be updated. A snapshot contains a copy of all the data in a query result, while a dynaset contains only a set of indirect references to the database records themselves. [Ref. 8]

There are both benefits and disadvantages to using a snapshot type Recordset. "A small Snapshot is generally faster to create and access than a Dynaset because its records are either in memory or stored in TEMP disk space and the Jet engine does not have to

deal with page locking or multi-user issues.”[Ref. 11] However, a Snapshot consumes more local memory.

3. Foreign Keys, Primary Keys, and Indexes

Once tables are created, and their fields, also referred to as columns, have been defined, three other features are used to assist with accessing the tables records and defining the relationships between the tables. These features are: Foreign Keys, Primary Keys, and Indexes.

Foreign keys are fields inserted into a table so that other tables can reference the records within it. These keys are called “foreign” because they are not obvious attributes of the tables in which they are contained. They are used specifically for other tables in order to query the table and select the records relevant to the calling table.

Primary keys are keys that make the record unique. The primary key may contain more than one field in order to make it unique. In the DODDS program, Dolphin-Id and Date are examples of fields that can make up the primary key of a table. In the Dolphin table, only the Dolphin-Id is required to make each record unique.

Most databases use indexes to access data faster. “Database table indexes are sorted lists that are faster to search than the actual tables.”[Ref. 3] A table may have multiple indexes defined to satisfy many different requirements of the program. For example, a report may be generated using the Dolphin table that shows all the names of the dolphins in alphabetical ascending order. Likewise, using the same table, a report may be generated in ascending order, showing the day that each dolphin was born.

B. TABLE MAINTENANCE CAPABILITIES

1. Building/Defining the Relational Tables

The ten relational tables and the elements contained within them can be found in Figure 1. The relational tables developed for the DODDS program are: Project,

Monthly_Stat, Constants, Video, Dolphin, Learned_Behavior, Daily_Stat, Session, Trainer, and Typical_Characteristics. Each table was constructed using Microsoft's Data Manager. Each table contains fields and records. Fields are also referred to as attributes or elements of a table. For each table, the primary key (which could be comprised of many fields), foreign keys, and any indexes are defined.

The Project table uses Project-Name and ID-Number as the primary key to make each record unique. The ID-Number, located in the Project table, is also the foreign key. The foreign key is used by the Dolphin table so that each project a dolphin has been involved with, can be retrieved when requested. The primary key is also used as the index in order to sort records by Project-Name, then by ID-Number.

The Monthly_Stat table's primary key consists of the ID-Number and Year-Month fields. These two fields combined make each record unique since statistics are developed once a month for each dolphin. The ID-Number, located in the Monthly_Stat table, is also the foreign key. The foreign key is used by the Dolphin table so that the monthly statistics of a dolphin can be retrieved on request. The primary key is also used as the index for sorting records by ID-Number and then by date in a YYMM format. The date format has been selected as YYMM to keep the records in a sorted order by year. MMY would scramble the records after multiple years of entering data.

The Constants table's primary key is ID-Number. This is sufficient to make each record unique since each dolphin will only have one Constants record associated with it. There is no foreign key in the Constants table and the ID-Number is used as the index for retrieving records.

The Video table's primary key is made up of the ID-Number and Training-Type fields. These two fields combined make each Video record unique. The ID-Number is considered the foreign key because it is not associated with a video, but is used by the Dolphin table to locate the appropriate records when requested. The records are indexed by ID-Number and Training-Type.

The Dolphin table has ID-Number as its primary key. There are no foreign keys for this table and records are indexed by the primary key. The Dolphin-ID is used in nine of ten tables for linking the tables.

The Learned_Behavior table uses the ID-Number and Behavior fields as its primary key. These two fields combined make each Learned_Behavior record unique. The ID-Number is also the foreign key and can be used by the Dolphin table to access certain dolphins' Behavioral records. The primary key is used as the index when records are retrieved from the table.

The Daily_Stat table uses the ID-Number and Date fields as its primary key. Since only one daily statistic is created per day for each dolphin, these two fields combined qualify as the primary key. The ID-Number is used as the foreign key because it is only required in the Daily_Stat table for locating the correct records within it by the Dolphin table. The primary key is used as the index when records are retrieved from the table.

The Session table uses four fields to make up the primary key. These four fields are ID-Number, Date, Session-Type, and Session-Type-Number. Multiple sessions are produced throughout the day and can include more than one session of the same type; therefore, all four fields are required to make up the primary key. ID-Number and Date make up the foreign key for the Daily_Stat table. These two fields are needed to access the correct dolphin's daily session records. Another foreign key defined in the Session table is Trainer-Initials. This field is used to link the Session table with the Trainer table. The Session table has two indexes. The first index is the same as the primary key, while the other is by the ID-Number field alone.

The Trainer table uses the Trainer-Initials as the primary key. This particular field was selected because each trainer puts his/her initials on the input forms, which can then be used to uniquely identify that individual. The field allows for three characters to help keep the trainer records unique. There is no foreign key for this table and records are indexed by the primary key.

The Typical_Characteristics table uses all five fields to make up the primary key. These fields are ID-Number, Date, Session-Type, Session-Type-Number, and Characteristic. These five fields combined keep the Typical_Characteristics records unique since multiple characteristics can be displayed in any one session. The ID-Number, Date, Session-Type, and Session-Type-Number fields are also used as the

foreign key to the Session table to enable Session records to access the characteristics of a dolphin displayed within a particular session.

2. Building/Defining the User-Defined Tables

The three user-defined tables and the elements contained within them can be found in Figure 1. The three user-defined tables developed for the DODDS program are the SessionTypes, SubTypes, and Characteristics tables. User-defined tables are used by the Progeny staff for adding, editing, and deleting records from within the pull down menus. These tables have been grouped differently because of their functionality. They are not related to any other tables and are unique within themselves. They are used only as the contents of drop-down lists within the DODDS program to allow for flexibility in the program as changes occur within the project. Each table was constructed using Microsoft's Data Manager, and can be accessed from within the program by means of the Data Control.

Currently, the SessionTypes table has four records which define the different types of behaviors a dolphin may perform. Not all activities must be performed in a single day. The session types consists of Training, Play, Enrichment, and Exercise. The majority of the sessions performed are of type Training. Allowing this to be a user-defined table enables the user to add or remove the types of sessions performed in the project. The session types show up on the daily input screens as drop-down lists when building the session records.

The SubTypes table contains the names of the behaviors that are being taught to the dolphins by the trainers. Dolphins can be trained particular behaviors or "Dolphins can learn by observation".[Ref. 12] These behaviors can be classified into three broad categories: Basic, Husbandry, and Systems. Basic behaviors include such elements as Bridging, Stationing, Targeting, Acoustic Recall, Gating, Separation, Dressing, and Habituation. Husbandry behaviors include Manipulation, Mouth/Eye Inspection, Exhalation, Probe Applications, Blood Sampling, Beaching, and Stretcher. Systems behaviors include behaviors such as Detection, Marker Delivery, Open Water Gating,

Open Water Control, Boat Follow, Open Water Beaching, Transport, and Pool Habituation. An explanation of each type of behaviors is beyond the scope of this thesis.

The Characteristics table has been provided to allow for populating and displaying possible dolphin characteristics. The possible characteristics are those that could be demonstrated by a dolphin during a session. Trainers populate this user-defined table to display in the DODDS program the more common characteristics shown by dolphins that the trainers want to track. The Characteristics table is dynamic in nature and will most likely require updating throughout the project. The Characteristics table is maintained throughout the duration of the Progeny project by the trainers. The table contains the particular characteristics that can be displayed and selected from a drop-down list. Selecting the characteristics will eventually create typical characteristic records that will be stored in the Typical_Characteristics table.

3. Database Integrity

The Data Control is an object provided by Visual Basic that is used for accessing the tables within the DODDS application. The Data Control has predefined events associated with it that are primarily for data integrity. The event names are; MoveFirst, MoveNext, MoveLast, MovePrevious, AddNew, Update, Delete, Find, Close and Bookmark. When referring to a set of records contained within a table, it is known as a Recordset.

The data control's Validate event allows you to check any changes made to the recordset before new information is written to the database. It also allows you to specify which record will become current after the Validate event is concluded. Validate is triggered when the current row is changed.[Ref. 3]

"The Validate event is invoked just before Visual Basic writes changes from the bound controls to the database and repositions the current record pointer to another row in the database."[Ref. 3] While the DODDS program does not use the built-in design structure, it does utilize the concepts for designing its own database integrity checks and record pointer checks.

Building tables that implement primary and foreign keys allows one to relate records in one table to those in another table.

For the relation to be useful, however, it is vital to maintain referential integrity when adding or deleting records. Referential integrity simply means that the foreign key(s) in any referencing table must always refer to a valid record in the referenced table.[Ref. 8]

4. Record Searching Capabilities

Three of the tables used by the DODDS program have an extra built-in feature to speed up the record searching capabilities for the users. These tables are Daily_Stat, Session, and Typical_Characteristics. The tables were selected to have this capability because of the sheer volume of records each could potentially possess. When accessing the tables, the user can locate the appropriate set of records by entering the ID-Number of the dolphin and the Date of the record for which he/she is searching. This will either take the user to the record directly, or to the set of records related to the search criteria.

C. DATA COLLECTION SCREENS

1. Two Primary Screens

Two primary input screens, the Monthly and Daily screens, were developed in the DODDS program to provide for monthly and daily data to be entered into their respective tables. Other pertinent data relative to the DODDS program is also entered into their respective tables, but more on an "as required" basis. For example, when a new video is created of a dolphin or a new trainer comes on board.

The Monthly input screen has input text boxes for each element of the Monthly_Stat table (Figure 7). The fields associated with the Monthly_Stat table are: ID-Number, Date (formatted as yymm), Blubber, Growth, and Weight. The Month_Stat table is accessed via Visual Basic's Data Control, which allows the users to pan through

the records in the table. Monthly_Stat records can be added, updated, and deleted from the input screen as well. Data integrity is enforced through enabling and disabling command buttons as the user manipulates the data. For example, the "Update" button is enabled only when a user actually changes a field's contents. The "Delete" button is disabled immediately after the "Add" button is pressed and is not enabled again until the "Update" button is pressed. Understanding where the actual record pointer is located within the table is essential in incorporating referential integrity in the Monthly_Stat table.

The screenshot shows a window titled "Monthly Statistics" containing a "Monthly Input Form". The form has five text input fields with labels: "ID_Number:" (containing "T1004F"), "Year_Month (yymm):" (containing "9612"), "Blubber (cm):" (containing "15"), "Growth (cm):" (containing "20"), and "Weight (lbs):" (containing "810"). Below these fields is a record navigation bar with left and right arrow buttons and the text "Record: 1". At the bottom are four buttons: "New Record", "Update", "Delete", and "Close". A status message at the very bottom reads "A Beep indicates a successful Update :^)".

Figure 7. Monthly Input Form.

The Daily input screens will be utilized most by the Progeny trainers. Two screens were created to handle the Daily input, the main screen and a continuation screen. Because the program allows for so many sessions to be input, a separate continuation screen needed to be developed. This second screen will be utilized mostly for the entry of history records where all the data is available for the entire day's activities. The first screen contains the input areas for the Daily record and for five Session records. The second screen allows for entering of 15 additional Session records at one time. The screens were designed using text boxes, drop-down lists, and buttons in order to accomplish various tasks.

Text boxes are used when the data is not constant or predictable. They provide the trainer a place to record information such as daily food intake in pounds, broken down by the different types of fish, total training time for the day that the trainer spent with the dolphins, temperature of the ocean water at the time of the training, and any other comments that may be noted during the day's activities (Figure 5).

Drop-down lists can be used for displaying the type of data that is relatively constant. The DODDS program uses drop-down lists for such things as selecting the dolphin for which the daily/session record is being recorded, the session type, the rating given by the trainer for a particular session, the actual behavior that was being taught, the primary trainer conducting the training exercise, and selecting typical characteristic(s) displayed during a session.

Buttons are used on the input forms for functions such as exiting the input screen, saving the daily/session data, viewing the pertinent tables without having to exit the input screens, viewing the pertinent data to facilitate in entering data, moving back and forth between the daily and continuation input screens, loading constant values into the appropriate fields for a particular dolphin, and calculating the Kilo-Calories given to a dolphin for the day.

2. Saving the Data

After the trainer has entered data into the daily input form, he/she has to push the "Save" button in order for the data showing on the form to be written to the appropriate tables. Once the "Save" button is selected, referential integrity checks and other tasks are performed by the program, such as checking for existing records, updating tables when appropriate, and displaying status messages and prompts to the user.

The first task that is performed by the "Save" procedure is to update the daily record that is currently entered on the input form. Before update of the daily record takes place, the record pointer is positioned to the beginning of the Daily_Stat table, and then a search of the table is performed. The purpose of the search is to determine if another record in the table having the same ID-Number and Date exists. If the search locates

another record that meets this criteria, the user has the option of overwriting the existing record and continuing on with the save procedure without saving the daily record, or canceling the save operation altogether.

The second task that is performed by the "Save" procedure is to prompt the user as to whether or not he/she desires to add any session records to the Session table. If the user responds "No", the procedure finishes. If the user responds "Yes", any session records located on the input forms are checked for accuracy ensuring all required fields have been filled in. If the session records do not pass this check, a message is displayed notifying the user of the problem and the "Save" procedure is exited.

If the session records pass the first validation check, each session record on the input form is checked against the Session table to determine if it already exists. If the session record is located in the table, the user is alerted and has the option of overwriting the existing session record or continuing on with any other session records entered on the input form. If the session record is not located, the session record is considered valid and is written to the Session table.

The last task performed by the "Save" routine is to add Typical Characteristics records associated with a session to the Typical_Characteristics table. Each session record has a check-box field that may or may not be checked by the user depending on whether or not there were any "Typical Characteristics" associated with the session. If the field has been marked, another pop-up form is displayed allowing the user to enter the typical characteristics displayed during the session (Figure 8). The typical characteristic(s) entered for the session is checked for uniqueness against the Typical_Characteristics table. If a duplicate record is found, the user is notified of the duplicate and is given the opportunity to discontinue processing of the typical characteristic records for the particular session or skipping the record in question. Bypassing the record allows processing to continue with any other typical characteristic(s) associated with the session record. If no record is located within the Typical_Characteristic table, the record is considered valid and is written to the table.

Session 1

Typical Characteristics

Seems Distracted

Diving deep down

Close

Figure 8. Typical Characteristic Input Form.

D. REPORT/GRAPH GENERATION

1. Using Crystal Reports

Visual Basic offers a report writing tool, Crystal Reports, which is used for the creation of sophisticated reports. The tool is accessible from the Visual Basic toolbar if full installation of the software is performed.

Crystal Reports is a powerful yet easy to use program for creating custom reports, lists, and form letters using data from your existing databases. The program works by establishing connections with one or more of your databases. Using these connections as conduits, Crystal Reports draws in the values from database fields you select and uses them in the report, either in their original form or as part of a formula that generates more sophisticated values.[Ref. 9]

The DODDS program uses Crystal Reports exclusively in generating the output reports. Each of the 13 tables contained in the DODDS application has a report generated for it so the data in the table can be displayed and printed in a readable and user-friendly format (Figure 6).

When developing a report through Crystal Reports, all that is required is to select the table(s) that contains the field(s) desired for displaying in the report. The next step is to select the field(s) from the table(s) and position them onto the report template provided. When data is selected from multiple tables, proper linking is required or an error message will be displayed.

Crystal Reports offer a variety of advanced features such as grouping data and creating summary and total information. Formulas and text fields can also be incorporated onto the forms. Reports can be created that will group records based on a change of any field contained in the report. Grouped records can give summary information such as maximums, minimums, or counts on the grouped records. Totals or grand totals can be produced per grouping or per report as well. Formulas can be used to provide more meaningful and complicated reports, and records can be displayed in ascending or descending order to suit user needs. Crystal Reports also allows for text fields to be incorporated on the reports to provide attractive and informative headings.

Reports can be designed during the development phase that either save the data so the reports do not change as new records are added to the tables, or that show the most current data located in the tables. Displaying the most current data in the tables is also known as "refreshing" the table data. The data is refreshed for all reports created within the DODDS application.

2. Record Sorting

Sorting records allows for reports to be generated that will display the records in a predetermined order or sort sequence. Sorts can be either ascending or descending. Ascending reports are in alphabetical or numerical order depending on the field selected. Descending sorts display records in the reverse order as an ascending report.

Reports are sorted based on the primary key of the table unless otherwise specified. If sorting via the primary key is not desirable, records can be sorted by any of the fields located on the report. The priority of sort fields is determined by the sequence of the fields selected during the creation of the report.

For the DODDS program, the Learned_Behavior, Constants, Daily_Stat, and Dolphin tables are all sorted by ID-Number. The Characteristics table is sorted by Characteristic, which is also the primary key. The Monthly_Stat table is sorted by ID-Number and Date. The Project table is sorted by Project-Name. The Session table is sorted by ID-Number, Date, Session-Type, and Session-Type-Number. The SessionTypes table is sorted by Session-Type. The SubType table is sorted by Subtype. The Trainer table is sorted by Trainer-Name. The Videos table is sorted by ID-Number and Training-Type. Lastly, the Typical_Characteristics table is sorted by ID-Number, Date, Session-Type, Session-Type-Number, and Characteristic. All reports generated are sorted in ascending order.

3. Using Visual Basic's Graph Utility

When deciding on which tool to use for the graphical output for the DODDS program, two alternatives were considered: the first choice was to export the data to another software package such as Microsoft Excel. The second choice was to use the graphics tool offered by Visual Basic.

Since Visual Basic offers the capability to plot Line Graphs using the data in the DODDS tables, the decision was made to use Visual Basic's features rather than rely on other applications. Using another application such as Excel would require creating a temporary file and then using the Object Linking Embedding (OLE) tools to link to the application for processing of the temporary file. This would also require the user to have this other application installed on his/her computer.

Visual Basic contains a feature called a Graph Control. "The graph control acts as a link between your application and the Graphics Server graphing and charting library." [Ref. 10] To utilize the graph capabilities of the Graphics Server, the graph

control is placed on the form as is any other object. The graph control has property values that can be set at either design time or run time. "This means that you can design a graph to whatever level of detail you need at design time, and then, at run time, set any or all properties to new values." [Ref. 10]

Data from the tables is stored in a graph property called GraphData. GraphData is a two-dimensional array. A data set is a collection of data supplied by one column in a table. Each individual element within a column is known as a *data point*.

Array properties are controlled through two properties: ThisSet and ThisPoint. ThisSet is the index for the data entered with the GraphData property. ThisPoint references the individual data points for the set specified by the ThisSet property. [Ref. 10]

E. SUMMARY

As evident from this chapter, tools such as the Data Manager, SQL, the Data Control, predefined events used to assist in data integrity, indexing and error trapping capabilities, the Graph Control, and Crystal Reports for output generation makes Microsoft's Visual Basic programming language the ideal product to develop extensive applications.

The next chapter discusses the output of the DODDS application. The output consists of reports and graphs. Each table defined in the database has a corresponding output report generated for it. Each report is discussed in detail as to the fields contained within it. The output graphs produced show relationships between selected variables and display the data in various layouts. Other output alternatives are briefly discussed.

V. PROGRAM OUTPUT

A. REPORTS AVAILABLE

The most basic output an application can produce is to display the contents of a table on the screen or route it to a printer to generate a hardcopy output of the same information. The DODDS program offers 13 reports that can be displayed on the screen and, if necessary, be printed out (Figure 6). Each table defined in the program has a report that has been created in order to display the records within the tables in some predefined order.

The report associated with the Learned_Behavior table shows what behaviors have been taught to which dolphins and the training time involved for accomplishing each training behavior. The report includes the ID number of the dolphin, the actual learned behavior, and the time it took to learn the behavior. Records are sorted and grouped based on ID number. Each group's total time for the learned behaviors that are listed is computed and included at the end of each grouping in the "Time To Learn" column. All the Learned_Behavior table records are included in the report.

The Characteristics table report shows the characteristics displayed by a dolphin during a session. The report includes characteristics contained in drop down lists that are available while building Typical Characteristic's records associated with each session. Records are sorted by Characteristic. There is no grouping of the records, and all records contained in the Characteristics table are included in the report.

The Constants table report displays the constants that are used to populate the daily statistics record. The report includes the ID number of the dolphin, the type of fish and amount, in pounds and Kilo-Calories, fed to the dolphin, and any vitamins given. Records are sorted by ID number and there is no grouping or totals within the report. All records contained in the Constants table are included in the report.

The report for the Daily_Stat table displays all the data associated with a dolphin for a particular day. The report includes all the input fields of the daily input record with the exception of the Comments field. The columns contained on the report include the ID number, date, weight and Kilo-Calories of each fish fed to the dolphin, amount of all

vitamins given, lost time, food intake, overall rating average, any medications given, total training time for the day, and water temperature of the Bay. Records are sorted by ID number and Date and are grouped by ID number. All records contained in the Daily_Stat table are included in the report.

The Dolphin table report lists all the dolphins that are part of the Progeny project. The report contains the ID number, name, and birth date of each dolphin. Records are sorted by ID number and the report does not contain any report groupings. All records in the Dolphin table are included in the report.

The Monthly_Stat table report shows the data associated with a dolphin's monthly physical evaluation. The report contains the ID number, date (expressed as YYMM), blubber, growth, and weight. These records are accumulated monthly for each dolphin. Records are sorted by the ID number and Date fields. Reports are grouped by the ID number of the dolphin and all records within the Monthly_Stat table are used in the report.

The report associated with the Project table displays those projects with which a particular dolphin has been involved. The report contains the Project Name, ID number of the dolphin, and beginning and ending dates. The dates signify the timeframe the dolphin has been considered a participant of a project. The records are sorted by the project name and dolphin's ID number. The report is grouped by the project name. All records in the Project table are included in the report.

The Session table report displays all the sessions conducted by the trainers for all dolphins. The report contains the ID number of the dolphin, session date, session type and number, length, rating, and specific trainer. Records are sorted by dolphin, date, session type, and session number. Records are grouped by ID number and all records are contained in the report.

The report for the SessionTypes table includes the session types as established by the trainers and Marine Mammal staff. The report is sorted by the session type and there is no grouping within the report. All records contained in the SessionType table are included in the report.

The SubTypes table report includes all the possible subtypes that can be performed within a session. A subtype is the name of the type of training behavior being performed during a session. Records are sorted by SubType and there is no grouping within the report. All records within the SubTypes table are included in the report.

The Trainer table report lists each of the trainers working in the Progeny project. It includes the trainer's name, date-of-birth, job title, and initials. Records are sorted by trainer name and there is no grouping within the report. All records in the Trainer table are included in the report.

The Typical_Characteristics table report shows the characteristics displayed by a dolphin during a session. The report includes the ID number of the dolphin, session date, session type, and session number, and the characteristic observed by the trainer. The records are sorted by ID number, date, session-type, session-number, and characteristic. There is no grouping within the report and all records in the table are included in the report.

The Video table report lists what videos have been taken of the dolphins. The videos are taken at the discretion of the trainer after a dolphin has proven itself proficient at a behavior. The Video table contains the dolphin ID, training type, tape creation date, and video filename. The records are sorted by the ID-number and training-type. Records are grouped by the ID number of the dolphin. All records contained in the Video table are included in the report.

B. GRAPH OUTPUT OPTIONS

Visual Basic offers graphic capabilities within an application by providing a Graph Control. "The graph control provides a range of graph types for displaying your data." [Ref. 10] The following graph types were implemented in the DODDS program: Area Graphs, Line Graphs, Bar Graphs (horizontal and vertical), and Stacked Bar Graphs (horizontal and vertical) (Figure 9).

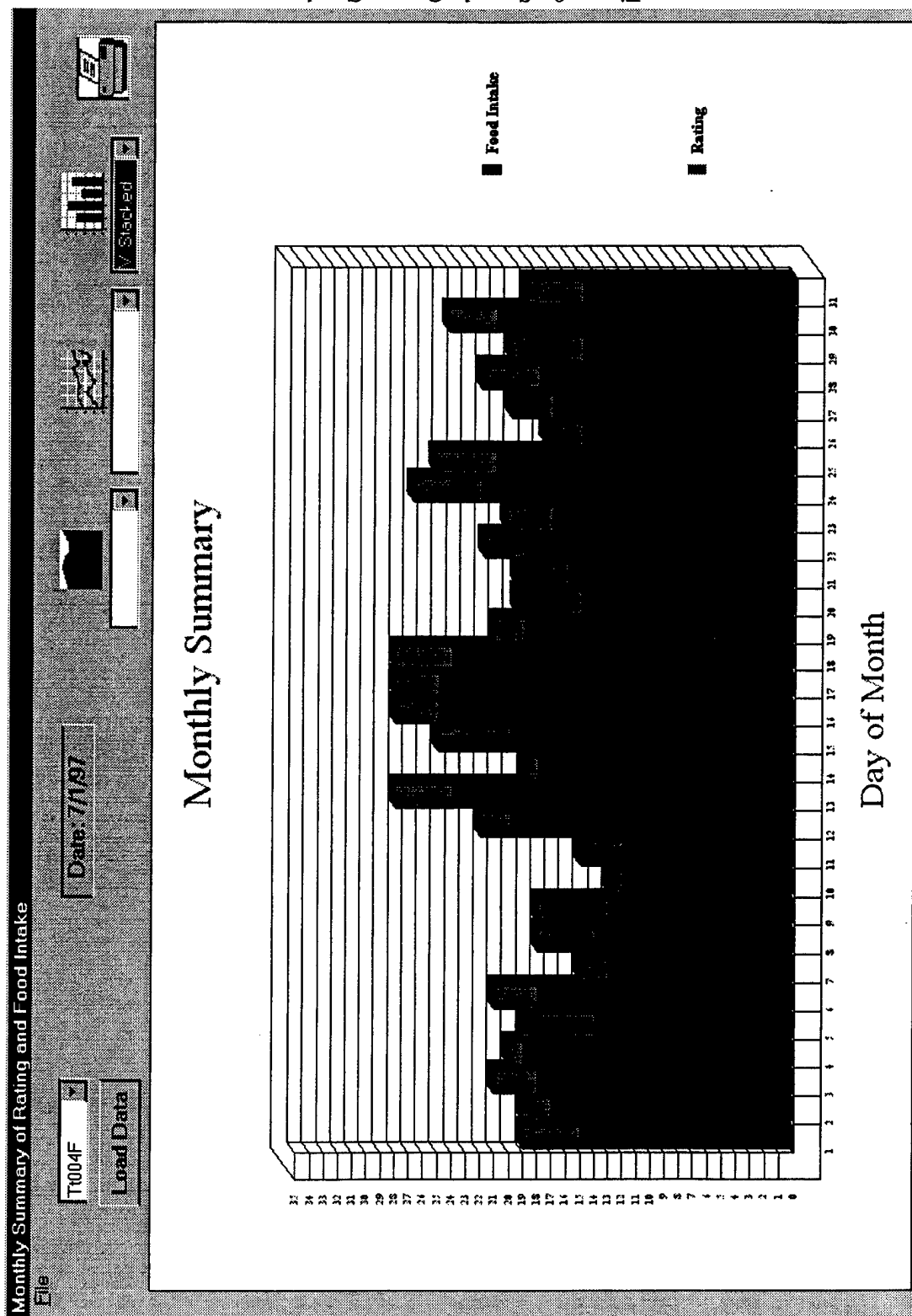


Figure 9. Sample Output Graph.

Not all graph types apply or are useful for displaying certain types of data, but the capabilities exist to allow for flexibility in displaying the data. The capability to use all graph types is provided to allow the trainers and staff to decide which graph type best suits their needs.

In the Progeny project, interpreting interaction in the data is very important in order for the dolphins to be trained in the most efficient means possible. Sometimes, unsuspected variables can influence the data. One example is the relationship between the food eaten by a dolphin and its weight. The logical assumption is that the more a dolphin is fed, the more rapidly the weight of the dolphin will increase. However, some relationships are not so obvious, such as the relationship of water temperature to vitamin deficiencies. It is because of this need to inspect multiple variables in the search for unique interactions that the graph outputs produced by the DODDS program have the capability to plot multiple data sets. Graphs can either be displayed on the screen or routed to a printer capable of handling graphics.

The DODDS program offers two graph outputs. The first graph option displays the relationship between a dolphin's blubber, weight, and growth, which is the data that is gathered on a monthly basis. The graph displays 12 months worth of data at a time. The second graph option displays the relationship between the amount of food given to a dolphin and the overall rating given for the dolphin's performance. The second graph's data is collected on a daily basis. The graph displays 30 days worth of data at a time.

C. OTHER OUTPUT ALTERNATIVES

Another output alternative for Progeny's data is to export the data to an entirely different application for further processing. For example, text files, made up of data from the tables, could be exported to Microsoft Excel to utilize the spreadsheet and graphic capabilities it provides. Likewise, other third-party software may have creative ways to manipulate the data that are different than those provided by Visual Basic's graphic utility.

D. SUMMARY

The DODDS application produces a report for each of the tables defined. Using Crystal Reports and indexing the tables is an easy way to create meaningful and informative reports. Crystal Reports sorts on the primary key for those tables where no indexes are defined. All fields within their respective tables are shown on each table's report.

The application also produces two graphs used to display the relationship between variables. One graph displays monthly data and contains three variables: blubber, weight, and growth. The other graph displays daily data and contains the average-rating and food-intake variables.

As with any application, improvements are always possible. In developing the DODDS application, various ideas came to mind as to ways in which this application could be improved. Improvement possibilities cover areas such as: record searching capabilities, more efficient ways to trap and handle errors, different ways to design the input screens, recording more data than is currently being recorded, and designing the system to work in a multi-user environment. The next chapter goes further into detail on these ideas and how one might implement them.

VI. CONCLUSIONS AND RECOMMENDATIONS

A. USER SATISFACTION

As the software product was being developed and prototypes were released in phases to the users, user satisfaction and user dissatisfaction became evident with each new release. As the product evolved, users were able to see their prior requirements fulfilled. Successfully implemented requirements always produced user satisfaction. However, as the product grew and updated prototypes were demonstrated to the users, more requirements would surface. Unfulfilled requirements often brought user dissatisfaction.

User satisfaction seemed most evident during the initial phases of development when the users could actually witness the development of their software product. Much excitement towards the software product was evident during the designing of the tables and seeing the first prototype actually running. It seemed to bring a sense of reality and satisfaction to their efforts put forth during the analysis and initial requirement building phase. User satisfaction was also apparent when previous requirements were demonstrated and proved to be working.

User dissatisfaction came in many varieties. Some of the dissatisfaction was because of failing hardware. Examples are: a video card problem that interfered during the prototype demonstrations, and a computer's internal speaker that was either not working or the sound was too low to hear message beeps put forth by the application. Dissatisfaction also surfaced while running older versions of software that tried unsuccessfully to access the data within the newer Visual Basic tables. New requirements that could not be implemented were also a source of dissatisfaction to the user.

One critical requirement that had to be resolved was the need for the trainers to view their previously entered session records (Figure 10). Since different trainers access the session input screen intermittently, the session data showing on the screen was usually another trainer's data. This forced the trainer to have to keep track of what session number was last conducted for the session type last performed. The easiest way a

Session Records

ID: Tt004F Date: 5/11/97

Session Type: E1	Rating: 2	Initials: MX	Time: 1	SubType: Bridging
Session Type: E2	Rating: 2	Initials: JR	Time: 3	SubType: Targeting
Session Type: E3	Rating: 1	Initials: MX	Time: 3	SubType: Stationing
Session Type: E4	Rating: 1	Initials: MX	Time: 4	SubType: Stationing
Session Type: E5	Rating: 2	Initials: MX	Time: 5	SubType: Bridging
Session Type: P1	Rating: 1	Initials: MX	Time: 2	SubType: Bridging
Session Type: P2	Rating: 2	Initials: JR	Time: 2	SubType: Bridging
Session Type: P3	Rating: 2	Initials: MX	Time: 3	SubType: Stationing
Session Type: PT1	Rating: 1	Initials: RB	Time: 11	SubType: Stationing
Session Type: T1	Rating: 2	Initials: RB	Time: 1	SubType: Bridging
Session Type: T2	Rating: 2	Initials: JR	Time: 2	SubType: Targeting
Session Type: T3	Rating: 2	Initials: MX	Time: 3	SubType: Stationing
Session Type: T4	Rating: 2	Initials: JR	Time: 4	SubType: Stationing
Session Type: T5	Rating: 2	Initials: MX	Time: 5	SubType: Stationing

Record Count is: 14

Figure 10. Session Records Form.

trainer could see their previously entered sessions, was to view the session table and search for their records. This seemed laborious and unfriendly. The solution was to have another button installed on the daily input screen that would produce a read-only report quickly showing all sessions entered for a dolphin on a particular day.

B. IMPROVEMENT POSSIBILITIES

As with any software product, there is always room for improvement. As the project was being developed, many ideas came to mind as to ways to improve the product. Currently, the application is fully operational as designed and developed. The following features though, describe enhancements that can be implemented to further improve the product.

1. Record Searching Techniques

As the data in the tables begin to grow, record searching techniques become important influences on the performance and ultimately, the user-friendliness of an application. When designing and developing the DODDS application, four basic options were available for searching or retrieving records from tables. These options were Seeks, Finds, Bookmarks, and scanning a table from Beginning Of File (BOF) to End Of File (EOF).

You use the Seek method to locate a record in a table-type recordset. To locate a record in a dynaset or snapshot, use one of the Find methods. When you use Seek to locate a record, Visual Basic uses the table's current index.[Ref. 8]

When retrieving a subset of data from a table, "...open the table directly, specify an index that has the data sorted on the criteria that you want to specify, and then seek to that location."[Ref. 8]

The Find method is good for locating a record of type dynaset or snapshot that meets a specific criteria within a recordset. "When trying to find an individual record, the best approach is to create a new recordset with a WHERE clause that contains the Find criteria."[Ref. 8] This will return all the records within the tables searched that match the criteria. Two criteria have to be met if the Find method is to be efficient. These criteria are:

The search criteria is a match against a single column which is indexed on the server. The search expression is either testing for equality or uses LIKE.[Ref. 8]

Terms such as WHERE and LIKE are used in SQL expressions. The WHERE clause is used to filter out specific records while scanning tables. The LIKE operator compares a string expression to a pattern looking for similarities between the two.

A Bookmark is another method provided by Visual Basic and is used for returning to a specific record within a table.

In Visual Basic, your view of records is usually a subset of the records in a table or combination of tables. Because the actual number of records can change at any time, especially in a multi-user environment, there's no absolute record number you can use to refer to a particular record. Instead, you can use bookmarks to identify, and then return to, a particular record.[Ref. 8]

Sometimes, during the execution of a program, individual records have to be recalled for subsequent processing of the application. The fastest way to return to a record is to store a bookmark and use it to return to that location. When considering the tradeoff between memory and execution time for using bookmarks, "The memory overhead in storing the string for the bookmark will be far less than the execution time involved in using a Find method to return to the record." [Ref. 8]

The last technique for retrieving records requires scanning the entire table in search of a record or records. This technique requires the most time to execute. It is used for searching a dynaset table for a record or records that are not indexed. Usually Beginning of File (BOF) and End of File (EOF) checks need to be performed on the table's records. "To cycle through all records, you can use the BOF and EOF properties to check for the beginning or end of the Dynaset." [Ref. 11] Programmatically, the EOF function would be used to detect the end of a table. The EOF function "...returns a value that indicates whether the end of a file has been reached." [Ref. 11]

The DODDS program uses Visual Basic's data control to locate records in the smaller tables. The data control allows the user to go to the BOF, EOF, or pan through the records one record at a time. All tables are defined as Dynaset because the program allows the records to be updated during program execution. When a form uses multiple tables for various functions, some tables are defined as table-type recordsets because they are only used for populating drop-down lists, and no updating of the table is permitted. An example of this would be when the dolphin table is used as a source for populating a drop-down list during the creation of daily records. When tables are accessed to only retrieve data, they are also defined as table-type recordsets because updating is not allowed. This condition also occurs when populating the daily input form with the constant data located in the Constants table. The Constants table has to be accessed to

retrieve data for the daily input form; however, no updates are allowed into the Constants table from the form.

Currently, the DODDS program checks for duplicate entries as part of a validation routine by scanning the entire recordset that it is accessing at the time. As the tables in the program begin to accumulate many records over time, this search technique will begin to slow down the response time, perhaps even to the point of becoming a major frustration to the user. The tables of concern are Daily_Stat, Session, and Typical_Characteristics. Each table is checked for duplicates upon entry of each new record as part of the validation routine for maintaining the integrity of the tables and the overall database design.

As a means of improving the program, it is suggested that the validation routines be changed to use the Find methods used on table-types of type Dynaset. Rather than scanning all the tables for duplicate entries for each new record that attempts to enter into the tables, the Find method should locate any duplicate records much more quickly. This will allow the DODDS program to continue running with acceptable response times regardless of the ultimate size of the tables.

2. Program Error Trapping

Visual Basic comes with another tool called the Data Form Designer. The Data Form Designer enables the developer to build input screens by supplying only the table name and field names. Most of the input screens in the DODDS program were initially built using the Data Form Designer tool. Each form was then altered to suit the program's needs.

The Data Form Designer tool builds the input form from a table's defined structure and supplies a data control used for cycling through the records within that table. Associated with the data control are procedures used for error-trapping and validation.

The Error procedure is where error handling code used for trapping errors is placed. Visual Basic automatically calls this procedure upon entering an error condition.

It is the developer's responsibility to enter code to handle program execution in the event of errors.

The Validation procedure is where validation code is placed for the following actions: MoveFirst, MovePrevious, MoveNext, MoveLast, AddNew, Update, Delete, Find, Bookmark, and Close. Errors can occur depending on the current position of the record pointer.

The DODDS program does not take advantage of the Error and Validation procedures defined within the data control object. Presently, all error trapping and validation is done in separate procedures normally associated with the button objects located on their respective forms.

An alternative way to code the DODDS program is to use the CASE structure supplied by the Validation procedure to centralize all error conditions in one procedure. This would ensure that the proper error routine procedure is called when a particular data control is being utilized. If a form has multiple data controls positioned on it, and the developer takes the responsibility for detecting error conditions, there is a chance of coding errors by the developer when attempting to reference the correct data control.

The DODDS program makes extensive use of one of the built-in functions provided by Visual Basic called "On Error". The On Error function is utilized by specifying at the beginning of a procedure where to proceed for error correction if one is encountered during the procedure's execution. If an error is encountered somewhere within a procedure, a Label is associated with the On Error condition. Execution of the code is passed to the Label for further processing of the error condition. A Label is nothing more than a starting point for further execution of the program code if an error condition occurs.

If the On Error condition is not present within a procedure that could fail, and an error does occur during execution, a default error message from Visual Basic is displayed and the program is halted. To avoid this situation, the On Error condition is utilized throughout the DODDS program.

3. Input Screens

Currently, the history data is recorded utilizing the same input forms in which the daily data is recorded. Since the same input form is used for both types of data, the program had to be designed and written to accommodate the maximum number of possible sessions per day that could appear on an old completed log sheet.

One way to improve the program would be to create a separate form for the history input and a separate form for the daily input. Since the trainers usually record only one or two sessions at a time, they do not require all the space allocated for the maximum number of sessions when recording their current daily and session data. Allowing for five sessions at a time would probably be sufficient. Another alternative is to have an input form that allows only one session at a time be input. The session data would then be entered into the database one record at a time.

Another improvement possibility would be to have a separate input form to save daily records apart from the session records. Since the daily data is recorded only once each day per dolphin, there is no reason to have that form also showing as the session data is being recorded throughout the day. In addition, it is likely that the data being reflected in the daily record will not be for the same dolphin as the session data being recorded. Displaying the daily record could confuse the trainer when inputting session data. Data on the daily record is not used when creating the session data record. Currently, there is the capability for the daily and session portions of the daily input screen to reflect a different dolphin and date.

4. Single-User vs. Networked System

Due to limitations of the Professional Edition version of the Visual Basic program, the DODDS program is used only on a single-user system. This allows the DODDS application to continue operating without taking into consideration the interaction of multiple users accessing the program and tables concurrently.

A networked system would be an advantage over the current system in order for the application, and the tables associated with it, to be accessed by multiple users for data

input. Reports could be produced independently by the Project manager, which would ensure that the data reflected is the most current since the Project manager would be accessing the same tables as the trainers. Another advantage would be that a secretary or student-aide could be entering historical data as the trainers access the program to fill in current data. Currently, the data resides on only one computer, therefore historical data has to be entered in a sequential manner along with the daily and session input.

5. Recording the Actual Trial Data

The data collected in the DODDS program, through the daily-input screens, records the summary data for each session. However, other data is also available during the course of a session that is currently not being recorded. During each session, a dolphin is put through a series of repetitive tasks to help the dolphin learn the behavior being taught. These repetitive tasks are known as *trials*. As a dolphin is being trained a certain behavior, the success or failure of each trial is recorded on a log sheet. It is from this log sheet that the trainer forms an opinion of the overall rating for the session. Only the rating for the session is recorded in the DODDS application, not the actual trial information.

One possible improvement to the DODDS program is to record the trial data produced during a session. From this data, graphical output could be produced that perhaps reflects patterns useful for the trainers, such as attention spans of a dolphin. Reports could be generated using calculations that show ratios and percentages of correct to incorrect responses. These ratios could then be plotted showing the improvement pattern as a dolphin learns a certain behavior.

As evident in this chapter of this thesis, many improvements could be made to the DODDS application that would aid in the data gathering and information generation for the Progeny project. It is the author's hope that these recommendations are someday implemented so that the DODDS application will continue to be a valuable asset and tool for the Marine Mammal Program here at NRaD.

APPENDIX. DODDS SOURCE CODE

**** PROGENY PROJECT.VBP ****

**** Visual Basic 4.0 ****

THIS SOURCE CODE IS CREATED AUTOMATICALLY WHEN A VISUAL BASIC APPLICATION IS SAVED AS A PROJECT IN VISUAL BASIC. IT LISTS EACH FORM USED WITHIN THE PROGRAM AND OTHER PARAMETERS AUTOMATICALLY INSERTED AND USED BY VISUAL BASIC. THE SOURCE CODE FOR EACH FORM LISTED IS INCLUDED IN THIS DOCUMENT. THE FORMS ARE LISTED ALPHABETICALLY.

Form=Charact.frm
Form=Constant.frm
Form=Daily.FRM
Form=DailyInp.FRM
Form=DailyInputContForm.frm
Form=Dolphin ViewChange Form.Frm
Form=Graphics.frm
Form=Learn.FRM
Form=MaintenanceForm.frm
Form=MonthlyInputForm.frm
Form=MonthlySummary.Frm
Form=OutputForm.frm
Form=progeny Form.Frm
Form=Project.frm
Form=Session#CharForm.Frm
Form=SessionRecordsForm.frm
Form=Sessions.FRM
Form=SessionT.FRM
Form=Status.frm
Form=SubTypes.FRM
Form=Trainers.FRM
Form=Typical.FRM
Form=Videos.FRM
Form=VideosTaken.frm

**** CHARACT.FRM ****

THIS SOURCE CODE DEFINES THE CHARACTERISTICS FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO ACCESS THE USER DEFINED TABLE NAMED *CHARACTERISTICS*.

```
Begin VB.Form frmCharact
Attribute VB_Name = "frmCharact"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Private Sub cmdAdd_Click()
    Data1.Recordset.AddNew
End Sub

Private Sub cmdDelete_Click()
Dim Response As Integer
    Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
        "Are you sure you want to do this?", vbYesNo)
    If Response = vbYes Then
        If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
            Data1.Recordset.Delete
            Data1.Recordset.MoveNext
            frmCharact.cmdUpdate.Enabled = False
            frmCharact.cmdNew.Enabled = True
            If Data1.Recordset.EOF = True Then
                Data1.Recordset.MovePrevious
            End If
        Else
            MsgBox "There are no records to delete."
        End If
    End If
End Sub

Private Sub cmdNew_Click()
    If Data1.Recordset.RecordCount = 0 Then
        txtFields(0).Enabled = True
    End If
    Data1.Recordset.AddNew
    frmCharact.cmdUpdate.Enabled = True
    frmCharact.cmdDelete.Enabled = False
End Sub

Private Sub cmdUpdate_Click()
    On Error GoTo ErrorHandler
    If Left(frmCharact.txtFields(0), 1) <> "" And _
        Left(frmCharact.txtFields(0), 1) <> " " Then
        Data1.UpdateRecord
        Data1.Recordset.Bookmark = Data1.Recordset.LastModified
        Beep
        frmCharact.cmdNew.Enabled = True
        frmCharact.cmdDelete.Enabled = True
    End If
Exit Sub
ErrorHandler:
```

```

MsgBox "Duplicate Record. No new record created."
frmCharact.cmdNew.Enabled = True
frmCharact.cmdDelete.Enabled = True
Data1.Recordset.MoveFirst
End Sub

```

```

Private Sub cmdClose_Click()
    frmCharact.cmdUpdate.Enabled = True
    MaintenanceForm.Visible = True
    Unload Me
End Sub

```

```

Private Sub Data1_Reposition()
    frmCharact.cmdUpdate.Enabled = False
    frmCharact.cmdNew.Enabled = True
    On Error Resume Next
    Data1.Caption = "Record: " & (Data1.Recordset.AbsolutePosition + 1)
End Sub

```

```

Private Sub Form_Activate()
    frmCharact.cmdNew.Enabled = True
    If Data1.Recordset.RecordCount = 0 Then
        txtFields(0).Enabled = False
    End If
End Sub

```

```

Private Sub txtFields_Change(Index As Integer)
    frmCharact.cmdUpdate.Enabled = True
    frmCharact.cmdNew.Enabled = False
End Sub

```

**** CONSTANT.FRM ****

THIS SOURCE CODE DEFINES THE CONSTANTS FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO ACCESS THE TABLE NAMED *CONSTANTS*.

```

Begin VB.Form frmConstants
Attribute VB_Name = "frmConstants"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

```

```

Private Sub cmdAdd_Click()
Dim i As Integer
If Data1.Recordset.RecordCount = 0 Then
    For i = 0 To 13
        txtFields(i).Enabled = True
    Next
End If
Data1.Recordset.AddNew
VideosViewChangeForm.cmdUpdate.Enabled = True
VideosViewChangeForm.cmdDelete.Enabled = False
End Sub

```

```

Private Sub cmdDelete_Click()
Dim Response As Integer
Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
    "Are you sure you want to do this?", vbYesNo)
If Response = vbYes Then
    If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
        Data1.Recordset.Delete
        Data1.Recordset.MoveNext
        frmConstants.cmdUpdate.Enabled = False
        frmConstants.cmdNew.Enabled = True
        If Data1.Recordset.EOF = True Then
            Data1.Recordset.MovePrevious
        End If
    Else
        MsgBox "There are no records to delete."
    End If
End If
End Sub

```

```

Private Sub cmdNew_Click()
Dim i As Integer
If Data1.Recordset.RecordCount = 0 Then
    For i = 0 To 13
        txtFields(i).Enabled = True
    Next
End If
Data1.Recordset.AddNew
frmConstants.cmdUpdate.Enabled = True
frmConstants.cmdDelete.Enabled = False
End Sub

```

```

Private Sub cmdUpdate_Click()
On Error GoTo ErrorHandler
If Left(frmConstants.txtFields(0), 1) <> "" And _
    Left(frmConstants.txtFields(0), 1) <> " " And _
    Left(frmConstants.txtFields(2), 1) <> "" And _
    Left(frmConstants.txtFields(2), 1) <> " " And _
    Left(frmConstants.txtFields(3), 1) <> "" And _
    Left(frmConstants.txtFields(3), 1) <> " " And _
    Left(frmConstants.txtFields(4), 1) <> "" And _
    Left(frmConstants.txtFields(4), 1) <> " " And _
    Left(frmConstants.txtFields(11), 1) <> "" And _
    Left(frmConstants.txtFields(11), 1) <> " " Then
    Data1.UpdateRecord
    Data1.Recordset.Bookmark = Data1.Recordset.LastModified
    Beep
    frmConstants.cmdNew.Enabled = True
    frmConstants.cmdDelete.Enabled = True
End If
Exit Sub
ErrorHandler:
MsgBox "Duplicate Record. No new record created."
frmConstants.cmdNew.Enabled = True
frmConstants.cmdDelete.Enabled = True
Data1.Recordset.MoveFirst

```

End Sub

```
Private Sub cmdClose_Click()  
    If MaintenanceForm.Tag = "Triggered by Maintenance Form" Then  
        MaintenanceForm.Visible = True  
        MaintenanceForm.Tag = ""  
    Else  
        DailyInputForm.Visible = True  
        frmConstants.cmdDelete.Enabled = True  
    End If  
    frmConstants.cmdUpdate.Enabled = True  
    Unload frmConstants  
End Sub
```

```
Private Sub Data1_Reposition()  
    frmConstants.cmdUpdate.Enabled = False  
    frmConstants.cmdNew.Enabled = True  
    On Error Resume Next  
    Data1.Caption = "Record: " & (Data1.Recordset.AbsolutePosition + 1)  
End Sub
```

```
Private Sub Form_Activate()  
    Dim i As Integer  
    frmConstants.cmdNew.Enabled = True  
    If Data1.Recordset.RecordCount = 0 Then  
        For i = 0 To 13  
            txtFields(i).Enabled = False  
        Next  
    End If  
End Sub
```

```
Private Sub Form_Load()  
    Dim today As Date  
    If DailyInputForm.Tag = "From Daily Input Form" Then  
        frmConstants.cmdDelete.Enabled = False  
        DailyInputForm.Tag = ""  
    End If  
End Sub
```

```
Private Sub txtFields_Change(Index As Integer)  
    frmConstants.cmdUpdate.Enabled = True  
    frmConstants.cmdNew.Enabled = False  
End Sub
```

**** DAILY.FRM ****

THIS SOURCE CODE DEFINES THE DAILY FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO ACCESS THE TABLE NAMED *DAILY_STAT*.

```
Begin VB.Form frmDaily  
    Attribute VB_Name = "frmDaily"  
    Attribute VB_Creatable = False
```

```
Attribute VB_Exposed = False
Option Explicit
```

```
Private Sub cmdAdd_Click()
    Data1.Recordset.AddNew
End Sub
```

```
Private Sub cmdDelete_Click()
Dim Response As Integer
    Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
        "Are you sure you want to do this?", vbYesNo)
    If Response = vbYes Then
        If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
            Data1.Recordset.Delete
            Data1.Recordset.MoveNext
            frmDaily.cmdUpdate.Enabled = False
            frmDaily.cmdNew.Enabled = True
            If Data1.Recordset.EOF = True Then
                Data1.Recordset.MovePrevious
            End If
        Else
            MsgBox "There are no records to delete."
        End If
    End If
End Sub
```

```
Private Sub cmdNew_Click()
Dim i As Integer
    If Data1.Recordset.RecordCount = 0 Then
        For i = 0 To 19
            txtFields(i).Enabled = True
        Next
    End If
    Data1.Recordset.AddNew
    frmDaily.cmdUpdate.Enabled = True
    frmDaily.cmdDelete.Enabled = False
End Sub
```

```
Private Sub cmdUpdate_Click()
On Error GoTo ErrorHandler
    If Left(frmDaily.txtFields(0), 1) <> "" And _
        Left(frmDaily.txtFields(0), 1) <> " " And _
        Left(frmDaily.txtFields(1), 1) <> "" And _
        Left(frmDaily.txtFields(1), 1) <> " " And _
        Left(frmDaily.txtFields(4), 1) <> "" And _
        Left(frmDaily.txtFields(4), 1) <> " " And _
        Left(frmDaily.txtFields(5), 1) <> "" And _
        Left(frmDaily.txtFields(5), 1) <> " " Then
        Data1.UpdateRecord
        Data1.Recordset.Bookmark = Data1.Recordset.LastModified
        Beep
        frmDaily.cmdNew.Enabled = True
        If MaintenanceForm.Tag = "Triggered by Maintenance Form" Then
            frmDaily.cmdDelete.Enabled = True
        End If
    End If
End Sub
```

```

Exit Sub
ErrorHandler:
MsgBox "Duplicate Record. No new record created."
frmDaily.cmdNew.Enabled = True
If MaintenanceForm.Tag = "Triggered by Maintenance Form" Then
    frmDaily.cmdDelete.Enabled = True
End If
Data1.Recordset.MoveFirst
End Sub

Private Sub cmdClose_Click()
If MaintenanceForm.Tag = "Triggered by Maintenance Form" Then
    MaintenanceForm.Visible = True
    MaintenanceForm.Tag = ""
Else
    DailyInputForm.Visible = True
End If
Unload frmDaily
End Sub

Private Sub Command5_Click()
Dim MyCriteria As String
Dim DolphinID As String * 10
Dim SessionDate As String * 8
DolphinID = Trim(DBCombo1.Text)
SessionDate = Trim(Text2.Text)
MyCriteria = "(ID_Number = " & "" & DolphinID & ")" & " AND " & "(CStr(Date) = " & "" &
SessionDate & ")"
frmDaily.Data1.Recordset.FindFirst MyCriteria
If Data1.Recordset.NoMatch Then
    MsgBox "Record Not Located"
End If
End Sub

Private Sub Data1_Reposition()
frmDaily.cmdUpdate.Enabled = False
frmDaily.cmdNew.Enabled = True
On Error Resume Next
Data1.Caption = "Record: " & (Data1.Recordset.AbsolutePosition + 1)
End Sub

Private Sub Form_Activate()
Dim i As Integer
frmDaily.cmdNew.Enabled = True
If Data1.Recordset.RecordCount = 0 Then
    For i = 0 To 19
        txtFields(i).Enabled = False
    Next
End If
End Sub

Private Sub Form_Load()
Dim today As Date
today = Now
today = Format(today, "mm/dd/yy")
Text2.Text = today

```

```

If DailyInputForm.Tag = "From Daily Input Form" Then
    frmDaily.cmdDelete.Enabled = False
    DailyInputForm.Tag = ""
End If
End Sub

```

```

Private Sub txtFields_Change(Index As Integer)
    frmDaily.cmdUpdate.Enabled = True
    frmDaily.cmdNew.Enabled = False
End Sub

```

**** DAILYINP.FRM ****

THIS SOURCE CODE DEFINES THE DAILY INPUT FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO ENTER DAILY AND SESSION RECORDS.

```

Begin VB.Form DailyInputForm
Attribute VB_Name = "DailyInputForm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

```

```

Public Sub Build1stSessionRecord()
    Data3.Recordset.Fields("ID_Number") = DBCombo17.Text
    Data3.Recordset.Fields("Date") = Trim(txtFields(0).Text)
    Data3.Recordset.Fields("Session_Type") = DBCombo7.Text
    Data3.Recordset.Fields("Session_Type_Number") = Trim(Text1(30).Text)
    Data3.Recordset.Fields("SubType") = DBCombo2.Text
    Data3.Recordset.Fields("Time") = Trim(Text1(31).Text)
    Data3.Recordset.Fields("Rating") = Combo1(0).Text
    Data3.Recordset.Fields("Trainer_Initials") = DBCombo12.Text
    Data3.UpdateRecord
    Data3.Recordset.Bookmark = Data3.Recordset.LastModified
End Sub

```

```

Public Sub Build2ndSessionRecord()
    Data3.Recordset.Fields("ID_Number") = DBCombo17.Text
    Data3.Recordset.Fields("Date") = Trim(txtFields(0).Text)
    Data3.Recordset.Fields("Session_Type") = DBCombo8.Text
    Data3.Recordset.Fields("Session_Type_Number") = Trim(Text1(32).Text)
    Data3.Recordset.Fields("SubType") = DBCombo3.Text
    Data3.Recordset.Fields("Time") = Trim(Text1(33).Text)
    Data3.Recordset.Fields("Rating") = Combo1(1).Text
    Data3.Recordset.Fields("Trainer_Initials") = DBCombo13.Text
    Data3.UpdateRecord
    Data3.Recordset.Bookmark = Data3.Recordset.LastModified
End Sub

```

THE SOURCE CODE CONTINUES THE SAME WAY FOR THE REMAINING 18 SESSION RECORDS. JUST THE TEXT AND COMBO FIELD NUMBERS CHANGE.

```

Public Sub AddOrEdit(RecordFound As Boolean)
    If RecordFound = False Then

```

```

        Data3.Recordset.AddNew
    Else
        Data3.Recordset.Edit
    End If
End Sub

Public Sub BuildSession1TypCharRcds()
    Dim j As Integer
    Dim RecordFound As Boolean
    Dim Response As Integer
    If DailyInputForm.Check1.Value = 1 Then
        frmStatus.lblStatus.Caption = "Processing Typical Char. Rcds."
        For j = 0 To 9
            If (Left(Session1CharForm.DBCombol(j), 1) <> "") And (Left(Session1CharForm.DBCombol(j), 1)
<> " ") Then
                RecordFound = False
                Call CheckTypicalRecordPointer
                Do While Data8.Recordset.EOF = False
                    If (Data8.Recordset.Fields("ID_Number") = DBCombol7.Text) And
(Trim(Data8.Recordset.Fields("Date")) = Trim(txtFields(0).Text)) And _
(Data8.Recordset.Fields("Session_Type") = DBCombo7.Text) And
(Trim(Data8.Recordset.Fields("Session_Type_Number")) = Trim(Text1(30).Text)) And _
(Trim(Data8.Recordset.Fields("Characteristic")) = Trim(Session1CharForm.DBCombol(j))) Then
                        Response = MsgBox("Typical Characteristics Record " & j + 1 & " already exists, Continue
processing Typical records?", vbYesNo)
                        RecordFound = True
                        If Response = vbNo Then
                            Data8.Recordset.MoveFirst
                            Exit Sub
                        Else
                            GoTo CheckNextCharacteristic
                        End If
                    End If
                    Data8.Recordset.MoveNext
                Loop
                If RecordFound = True Then
                    Data8.Recordset.Edit
                Else
                    Data8.Recordset.AddNew
                End If
                Data8.Recordset.Fields("ID_Number") = DBCombol7.Text
                Data8.Recordset.Fields("Date") = Trim(txtFields(0).Text)
                Data8.Recordset.Fields("Session_Type") = DBCombo7.Text
                Data8.Recordset.Fields("Session_Type_Number") = Trim(Text1(30).Text)
                Data8.Recordset.Fields("Characteristic") = Trim(Session1CharForm.DBCombol(j))
                Data8.UpdateRecord
                Data8.Recordset.Bookmark = Data8.Recordset.LastModified
            End If
        CheckNextCharacteristic:
        Next
    End If
End Sub

Public Sub BuildSession2TypCharRcds()
    Dim j As Integer
    Dim RecordFound As Boolean

```

```

Dim Response As Integer
If DailyInputForm.Check2.Value = 1 Then
    frmStatus.lblStatus.Caption = "Processing Typical Char. Rcds."
    For j = 0 To 9
        If (Left(Session2CharForm.DBCombol(j), 1) <> "") And (Left(Session2CharForm.DBCombol(j), 1)
        <> " ") Then
            RecordFound = False
            Call CheckTypicalRecordPointer
            Do While Data8.Recordset.EOF = False
                If (Data8.Recordset.Fields("ID_Number") = DBCombol7.Text) And
                (Trim(Data8.Recordset.Fields("Date")) = Trim(txtFields(0).Text)) And _
                (Data8.Recordset.Fields("Session_Type") = DBCombo8.Text) And
                (Trim(Data8.Recordset.Fields("Session_Type_Number")) = Trim(Text1(32).Text)) And _
                (Trim(Data8.Recordset.Fields("Characteristic")) = Trim(Session2CharForm.DBCombol(j))) Then
                    Response = MsgBox("Typical Characteristics Record " & j + 1 & " already exists, Continue
                    processing Typical records?", vbYesNo)
                    RecordFound = True
                    If Response = vbNo Then
                        Data8.Recordset.MoveFirst
                        Exit Sub
                    Else
                        GoTo CheckNextCharacteristic
                    End If
                End If
                Data8.Recordset.MoveNext
            Loop
            If RecordFound = True Then
                Data8.Recordset.Edit
            Else
                Data8.Recordset.AddNew
            End If
            Data8.Recordset.Fields("ID_Number") = DBCombol7.Text
            Data8.Recordset.Fields("Date") = Trim(txtFields(0).Text)
            Data8.Recordset.Fields("Session_Type") = DBCombo8.Text
            Data8.Recordset.Fields("Session_Type_Number") = Trim(Text1(32).Text)
            Data8.Recordset.Fields("Characteristic") = Trim(Session2CharForm.DBCombol(j))
            Data8.UpdateRecord
            Data8.Recordset.Bookmark = Data8.Recordset.LastModified
        End If
    CheckNextCharacteristic:
        Next
    End If
End Sub

```

THE SOURCE CODE CONTINUES THE SAME WAY FOR THE REMAINING 18 SESSION RECORDS. JUST THE TEXT AND COMBO FIELD NUMBERS CHANGE.

```

Public Sub CheckSessionRecordPointer()
    If Data3.Recordset.BOF = False Then
        Data3.Recordset.MoveFirst
    End If
End Sub

```

```

Public Sub CheckTypicalRecordPointer()
    If Data8.Recordset.BOF = False Then
        Data8.Recordset.MoveFirst
    End If
End Sub

```

```
End If
End Sub
```

```
Private Sub cmdDelete_Click()
    'this may produce an error if you delete the last
    'record or the only record in the recordset
    Data1.Recordset.Delete
    Data1.Recordset.MoveNext
    Data3.Recordset.Delete
    Data3.Recordset.MoveNext
End Sub
```

```
Public Sub ValidateSessionRecords(QuitUpdates As Boolean)
    frmStatus.lblStatus.Caption = "Validating Session Records...."
    If DBCombo7.Text <> "" Then
        If (Trim(Text1(30).Text) = "") Or (DBCombo2.Text = "") Or (Trim(Text1(31).Text) = "") Or _
            (Combo1(0).Text = "") Or (DBCombo12.Text = "") Then
            MsgBox "All session #1 fields must be entered. Please fix and re-save records."
            QuitUpdates = True
            Exit Sub
        End If
    End If
    If DBCombo8.Text <> "" Then
        If (Trim(Text1(32).Text) = "") Or (DBCombo3.Text = "") Or (Trim(Text1(33).Text) = "") Or _
            (Combo1(1).Text = "") Or (DBCombo13.Text = "") Then
            MsgBox "All session #2 fields must be entered. Please fix and re-save records."
            QuitUpdates = True
            Exit Sub
        End If
    End If
End Sub
```

THE SOURCE CODE CONTINUES THE SAME WAY FOR THE REMAINING 18 SESSION RECORDS. JUST THE TEXT AND COMBO FIELD NUMBERS CHANGE.

```
Private Sub Check1_Click()
    Load Session1CharForm
    Session1CharForm.Visible = True
    DailyInputForm.Visible = False
End Sub
```

```
Private Sub Check2_Click()
    Load Session2CharForm
    Session2CharForm.Visible = True
    DailyInputForm.Visible = False
End Sub
```

THE SOURCE CODE CONTINUES THE SAME WAY FOR THE REMAINING 18 SESSION RECORDS. JUST THE FORM NUMBERS CHANGE.

```
Private Sub cmdUpdate_Click()
    Dim RecordFound As Boolean
    Dim Response As Integer
    Dim QuitUpdates As Boolean
    Load frmStatus
    frmStatus.Visible = True
```

```

frmStatus.SetFocus
QuitUpdates = False
Call UpdateDailyStatistic(QuitUpdates)
If QuitUpdates = True Then
    Unload frmStatus
    Exit Sub
End If
frmStatus.lblStatus.Caption = " "
Response = MsgBox("Add/Update the Session records?", vbYesNo)
If Response = vbNo Then
    Unload frmStatus
    Exit Sub
End If
QuitUpdates = False
Call ValidateSessionRecords(QuitUpdates)
If QuitUpdates = True Then
    Unload frmStatus
    Exit Sub
End If
If DBCombo7.Text <> "" Then
    frmStatus.lblStatus.Caption = "Processing Session Records...."
    RecordFound = False
    Call CheckSessionRecordPointer
    Do While Data3.Recordset.EOF = False
        If (Data3.Recordset.Fields("ID_Number") = DBCombo17.Text) And (Data3.Recordset.Fields("Date")
= txtFields(0).Text) And _
        (Data3.Recordset.Fields("Session_Type") = DBCombo7.Text) And
(Data3.Recordset.Fields("Session_Type_Number") = Text1(30).Text) Then
            Response = MsgBox("Session 1 Record already exists, Continue updating Session 1 record?",
vbYesNoCancel)
            If Response = vbCancel Then
                Unload frmStatus
                Exit Sub
            ElseIf Response = vbNo Then
                GoTo Session2Start
            End If
            RecordFound = True
            GoTo Session1Update
        End If
        Data3.Recordset.MoveNext
    Loop
Else
    frmStatus.lblStatus.Caption = "0 Session Records processed. "
    frmStatus.Command1.Visible = True
    frmStatus.Command1.Enabled = True
    Exit Sub
End If

Session1Update:
Call AddOrEdit(RecordFound)
Call Build1stSessionRecord
If DailyInputForm.Check1.Value = 1 Then
    Call BuildSession1TypCharRcds
End If

```

Session2Start:

```

If DBCombo8.Text <> "" Then
    frmStatus.lblStatus.Caption = "Processing Session Records...."
    RecordFound = False
    Call CheckSessionRecordPointer
    Do While Data3.Recordset.EOF = False
        If (Data3.Recordset.Fields("ID_Number") = DBCombo17.Text) And (Data3.Recordset.Fields("Date")
= txtFields(0).Text) And _
            (Data3.Recordset.Fields("Session_Type") = DBCombo8.Text) And
(Data3.Recordset.Fields("Session_Type_Number") = Text1(32).Text) Then
            Response = MsgBox("Session 2 Record already exists, Continue updating Session 2 record?",
vbYesNoCancel)
            If Response = vbCancel Then
                Unload frmStatus
                Exit Sub
            ElseIf Response = vbNo Then
                GoTo Session3Start
            End If
            RecordFound = True
            GoTo Session2Update
        End If
        Data3.Recordset.MoveNext
    Loop
Else
    frmStatus.lblStatus.Caption = "1 Session Record processed. "
    frmStatus.Command1.Visible = True
    frmStatus.Command1.Enabled = True
    Exit Sub
End If

Session2Update:
    Call AddOrEdit(RecordFound)
    Call Build2ndSessionRecord
    If DailyInputForm.Check2.Value = 1 Then
        Call BuildSession2TypCharRcds
    End If

```

THE SOURCE CODE CONTINUES THE SAME WAY FOR THE REMAINING 18 SESSION RECORDS. JUST THE TEXT AND COMBO FIELD NUMBERS CHANGE.

```

Private Sub cmdClose_Click()
Dim Response As Integer
    Response = MsgBox("Do you need to save your data first?", vbYesNo)
    If Response = vbYes Then
        Exit Sub
    End If
    Unload frmStatus
    Unload frmConstants
    Unload SessionsViewChangeForm
    Unload frmDaily
    Unload TypicalViewChangeForm
    Unload DailyInputContForm
    Unload DailyInputForm
    Unload Session1CharForm
    Unload Session2CharForm
    ** CLOSE REMAINING 18 FORMS **
    MaintenanceForm.Visible = True

```

End Sub

Private Sub cmdViewRecords_Click()

DailyInputForm.Visible = False

Load SessionRecordsForm

SessionRecordsForm.Visible = True

SessionRecordsForm.SetFocus

End Sub

Private Sub Command1_Click()

DailyInputContForm.Visible = True

DailyInputContForm.Label35.Caption = DailyInputForm.DBCombo7.Text &
DailyInputForm.Text1(30).Text

DailyInputContForm.Label36.Caption = DailyInputForm.DBCombo8.Text &
DailyInputForm.Text1(32).Text

DailyInputContForm.Label37.Caption = DailyInputForm.DBCombo9.Text &
DailyInputForm.Text1(34).Text

DailyInputContForm.Label38.Caption = DailyInputForm.DBCombo10.Text &
DailyInputForm.Text1(36).Text

DailyInputContForm.Label39.Caption = DailyInputForm.DBCombo11.Text &
DailyInputForm.Text1(38).Text

DailyInputForm.Visible = False

End Sub

Private Sub Command2_Click()

Dim today As Date

Dim Response As Integer

Dim Total_Base_Lbs As Single

Dim Total_Base_KCals As Single

today = Now

today = Format(today, "mm/dd/yy")

Response = MsgBox("Load defaults for Session Input too?", vbYesNo)

If Response = vbYes Then

DBCombo17.Text = DBCombo1.Text

txtFields(0).Text = today

End If

txtFields(1).Text = today

DailyInputForm.txtFields(2).Text = 0

DailyInputForm.txtFields(3).Text = 0

DailyInputForm.txtFields(4).Text = 0

DailyInputForm.txtFields(5).Text = 0

DailyInputForm.txtFields(6).Text = 0

DailyInputForm.txtFields(7).Text = 0

DailyInputForm.txtFields(8).Text = 0

DailyInputForm.txtFields(9).Text = 0

DailyInputForm.txtFields(10).Text = 0

DailyInputForm.txtFields(11).Text = 0

DailyInputForm.txtFields(12).Text = 0

DailyInputForm.txtFields(13).Text = 0

DailyInputForm.txtFields(14).Text = 0

DailyInputForm.txtFields(15).Text = 0

DailyInputForm.txtFields(16).Text = 0

DailyInputForm.txtFields(18).Text = 0

DailyInputForm.txtFields(19).Text = " "

If Data7.Recordset.RecordCount <> 0 Then

Data7.Recordset.MoveFirst

```

End If
Do While Data7.Recordset.EOF = False
    If Data7.Recordset.Fields("ID_Number") = DBCombo1.Text Then
        DailyInputForm.txtFields(7).Text = Data7.Recordset.Fields("Herring_Lbs")
        DailyInputForm.txtFields(10).Text = Data7.Recordset.Fields("Pacific_Mackeral_Lbs")
        DailyInputForm.txtFields(2).Text = Data7.Recordset.Fields("Capelin_Lbs")
        DailyInputForm.txtFields(3).Text = Data7.Recordset.Fields("Columbia_River_Smelt_Lbs")
        DailyInputForm.txtFields(12).Text = Data7.Recordset.Fields("Squid_Lbs")
        DailyInputForm.txtFields(11).Text = Data7.Recordset.Fields("Sea_Tablets")
        DailyInputForm.txtFields(14).Text = Data7.Recordset.Fields("Vitamin_E")
        DailyInputForm.txtFields(13).Text = Data7.Recordset.Fields("Vitamin_C")
        Total_Base_Lbs = (Data7.Recordset.Fields("Herring_Lbs") +
Data7.Recordset.Fields("Pacific_Mackeral_Lbs") + _
        Data7.Recordset.Fields("Capelin_Lbs") + Data7.Recordset.Fields("Columbia_River_Smelt_Lbs") + _
        Data7.Recordset.Fields("Squid_Lbs"))
        DailyInputForm.txtFields(4).Text = Total_Base_Lbs
        DailyInputForm.txtFields(6).Text = Total_Base_Lbs
        Total_Base_KCals = ((Data7.Recordset.Fields("Herring_Lbs") *
Data7.Recordset.Fields("Herring_KCals")) + _
        (Data7.Recordset.Fields("Pacific_Mackeral_Lbs") * Data7.Recordset.Fields("Pac_Mack_KCals")) +
        -
        (Data7.Recordset.Fields("Capelin_Lbs") * Data7.Recordset.Fields("Capelin_KCals")) + _
        (Data7.Recordset.Fields("Columbia_River_Smelt_Lbs") *
Data7.Recordset.Fields("CRSmelt_KCals")) + _
        (Data7.Recordset.Fields("Squid_Lbs") * Data7.Recordset.Fields("Squid_KCals")))
        DailyInputForm.txtFields(5).Text = Total_Base_KCals
        DailyInputForm.txtFields(8).Text = Total_Base_KCals
        Exit Sub
    End If
    Data7.Recordset.MoveNext
Loop
MsgBox "Dolphin's 'Constant Record' not located."
End Sub

Private Sub Command3_Click()
    DailyInputForm.Tag = "From Daily Input Form"
    DailyInputForm.Visible = False
    Load SessionsViewChangeForm
    SessionsViewChangeForm.Visible = True
    SessionsViewChangeForm.SetFocus
End Sub

Private Sub Command4_Click()
    DailyInputForm.Tag = "From Daily Input Form"
    DailyInputForm.Visible = False
    Load frmDaily
    frmDaily.Visible = True
    frmDaily.SetFocus
End Sub

Private Sub Command5_Click()
    DailyInputForm.Tag = "From Daily Input Form"
    DailyInputForm.Visible = False
    Load TypicalViewChangeForm
    TypicalViewChangeForm.Visible = True
    TypicalViewChangeForm.SetFocus

```

End Sub

```
Private Sub Command6_Click()  
    DailyInputForm.Tag = "From Daily Input Form"  
    DailyInputForm.Visible = False  
    Load frmConstants  
    frmConstants.Visible = True  
    frmConstants.SetFocus  
End Sub
```

```
Private Sub Command7_Click()  
    Dim today As Date  
    today = Now  
    today = Format(today, "mm/dd/yy")  
    txtFields(0).Text = today  
End Sub
```

```
Private Sub Command8_Click()  
    Dim Total_Intake As Single  
    Dim Total_Intake_KCals As Single  
    If DailyInputForm.txtFields(7).Text = "" Or DailyInputForm.txtFields(7) = " " Then  
        DailyInputForm.txtFields(7).Text = 0  
    End If  
    If DailyInputForm.txtFields(10).Text = "" Or DailyInputForm.txtFields(10) = " " Then  
        DailyInputForm.txtFields(10).Text = 0  
    End If  
    If DailyInputForm.txtFields(2).Text = "" Or DailyInputForm.txtFields(2) = " " Then  
        DailyInputForm.txtFields(2).Text = 0  
    End If  
    If DailyInputForm.txtFields(3).Text = "" Or DailyInputForm.txtFields(3) = " " Then  
        DailyInputForm.txtFields(3).Text = 0  
    End If  
    If DailyInputForm.txtFields(12).Text = "" Or DailyInputForm.txtFields(12) = " " Then  
        DailyInputForm.txtFields(12).Text = 0  
    End If  
    Total_Intake = (CInt(DailyInputForm.txtFields(7)) + CInt(DailyInputForm.txtFields(10)) + _  
        CInt(DailyInputForm.txtFields(2)) + CInt(DailyInputForm.txtFields(3)) + _  
        CInt(DailyInputForm.txtFields(12)))  
    DailyInputForm.txtFields(6).Text = Total_Intake  
    If Data7.Recordset.RecordCount <> 0 Then  
        Data7.Recordset.MoveFirst  
    End If  
    Do While Data7.Recordset.EOF = False  
        If Data7.Recordset.Fields("ID_Number") = DBCombo1.Text Then  
            Total_Intake_KCals = ((Data7.Recordset.Fields("Herring_KCals") *  
CInt(DailyInputForm.txtFields(7))) + _  
                (Data7.Recordset.Fields("Pac_Mack_KCals") * CInt(DailyInputForm.txtFields(10))) + _  
                (Data7.Recordset.Fields("Capelin_KCals") * CInt(DailyInputForm.txtFields(2))) + _  
                (Data7.Recordset.Fields("CRSmelt_KCals") * CInt(DailyInputForm.txtFields(3))) + _  
                (Data7.Recordset.Fields("Squid_KCals") * CInt(DailyInputForm.txtFields(12))))  
            DailyInputForm.txtFields(8).Text = Total_Intake_KCals  
            Exit Sub  
        End If  
        Data7.Recordset.MoveNext  
    Loop  
    MsgBox "Dolphin's 'Constant Record' not located."
```

```
DailyInputForm.txtFields(8).Text = 0
End Sub
```

```
Private Sub Command9_Click()
Dim Response As Integer
Response = MsgBox("This will clear the Daily Input Form and both Session Forms!" & Chr(13) &
Chr(10) & _
"Are you sure you want to do this?", vbYesNo)
If Response = vbYes Then
MousePointer = vbHourglass
Unload frmStatus
Unload frmConstants
Unload SessionsViewChangeForm
Unload frmDaily
Unload TypicalViewChangeForm
Unload DailyInputContForm
Unload DailyInputForm
Unload Session1CharForm
Unload Session2CharForm
UNLOAD REMAINING 18 FORMS
Load DailyInputForm
Load DailyInputContForm
DailyInputForm.Visible = True
MousePointer = vbDefault
End If
End Sub
```

```
Private Sub Data1_Reposition()
On Error Resume Next
This will display the current record position
for dynasets and snapshots
Data1.Caption = "Record: " & (Data1.Recordset.AbsolutePosition + 1)
End Sub
```

```
Private Sub Data3_Reposition()
On Error Resume Next
This will display the current record position
for dynasets and snapshots
Data3.Caption = "Record: " & (Data3.Recordset.AbsolutePosition + 1)
End Sub
```

```
Private Sub Form_Load()
Dim i As Integer
For i = 0 To 4
Combo1(i).AddItem 1
Combo1(i).AddItem 2
Combo1(i).AddItem 3
Combo1(i).AddItem 4
Combo1(i).AddItem 5
Next
End Sub
```

```
Public Sub UpdateDailyStatistic(QuitUpdates As Boolean)
Dim RecordFound As Boolean
Dim Response As Integer
frmStatus.lblStatus.Caption = "Processing Daily Statistics..."
```

```

RecordFound = False
QuitUpdates = False
If Data1.Recordset.RecordCount <> 0 Then
    Data1.Recordset.MoveFirst
End If
Do While Data1.Recordset.EOF = False
    If Data1.Recordset.Fields("ID_Number") = DBCombo1.Text And Trim(Data1.Recordset.Fields("Date"))
= Trim(txtFields(1).Text) Then
        Response = MsgBox("Daily Stat record already exists, Overwrite existing data?", vbYesNoCancel)
        If Response = vbCancel Then
            QuitUpdates = True
            Exit Sub
        End If
        If Response = vbNo Then
            GoTo Finished
        End If
        RecordFound = True
        GoTo UpdateRecord
    End If
    Data1.Recordset.MoveNext
Loop

```

UpdateRecord:

```

If (txtFields(1).Text <> "") And (DBCombo1.Text <> "") Then
    If RecordFound = False Then
        Data1.Recordset.AddNew
    Else
        Data1.Recordset.Edit
    End If
    Data1.Recordset.Fields("ID_Number") = DBCombo1.Text
    Data1.Recordset.Fields("Date") = Trim(txtFields(1).Text)
    Data1.Recordset.Fields("Herring_Lbs") = Trim(txtFields(7).Text)
    Data1.Recordset.Fields("Pacific_Mackerel_Lbs") = Trim(txtFields(10).Text)
    Data1.Recordset.Fields("Capelin_Lbs") = Trim(txtFields(2).Text)
    Data1.Recordset.Fields("Columbia_River_Smelt_Lbs") = Trim(txtFields(3).Text)
    Data1.Recordset.Fields("Squid_Lbs") = Trim(txtFields(12).Text)
    Data1.Recordset.Fields("Food_Intake") = Trim(txtFields(6).Text)
    Data1.Recordset.Fields("Food_Intake_KCals") = Trim(txtFields(8).Text)
    Data1.Recordset.Fields("Special_Meds") = chkFields(17).Value
    Data1.Recordset.Fields("Sea_Tablets") = Trim(txtFields(11).Text)
    Data1.Recordset.Fields("Vitamin_E") = Trim(txtFields(14).Text)
    Data1.Recordset.Fields("Vitamin_C") = Trim(txtFields(13).Text)
    Data1.Recordset.Fields("Food_Base_Lbs") = Trim(txtFields(4).Text)
    Data1.Recordset.Fields("Food_Base_KCals") = Trim(txtFields(5).Text)
    Data1.Recordset.Fields("Lost_Time") = Trim(txtFields(9).Text)
    Data1.Recordset.Fields("Total_Training_Time") = Trim(txtFields(18).Text)
    Data1.Recordset.Fields("Water_Temp") = Trim(txtFields(15).Text)
    Data1.Recordset.Fields("Rating_Average") = Trim(txtFields(16).Text)
    Data1.Recordset.Fields("Comments") = txtFields(19).Text
    Data1.UpdateRecord
    Data1.Recordset.Bookmark = Data1.Recordset.LastModified
    MsgBox "Daily Stat. record has been saved"
Else
    MsgBox "Date and ID_Number are both required fields"
    Exit Sub
End If

```

Finished:End Sub

**** DAILYINPCONTFORM.FRM ****

THIS SOURCE CODE DEFINES THE DAILY INPUT CONTINUATION FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO ENTER SESSION RECORDS 6 TO 20.

```
Begin VB.Form DailyInputContForm
Attribute VB_Name = "DailyInputContForm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit
```

```
Private Sub Check6_Click()
Load Session6CharForm
Session6CharForm.Visible = True
DailyInputContForm.Visible = False
End Sub
```

```
Private Sub Check7_Click()
Load Session7CharForm
Session7CharForm.Visible = True
DailyInputContForm.Visible = False
End Sub
```

THIS CONTINUES FOR THE REMAINING 13 FORMS ON THE DAILY CONTINUATION PAGE.

```
Private Sub Command1_Click()
DailyInputContForm.Visible = False
DailyInputForm.Visible = True
End Sub
```

```
Private Sub Form_Load()
Dim i As Integer
For i = 5 To 19
Combo1(i).AddItem 1
Combo1(i).AddItem 2
Combo1(i).AddItem 3
Combo1(i).AddItem 4
Combo1(i).AddItem 5
Next
End Sub
```

**** DOLPHINVIEWCHANGEFORM.FRM ****

THIS SOURCE CODE DEFINES THE DOLPHINS FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO ACCESS THE TABLE NAMED *DOLPHIN*.

VERSION 4.00

```
Begin VB.Form DolphinViewChangeForm
```

```

Attribute VB_Name = "DolphinViewChangeForm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

```

```

Private Sub Command1_Click()
    Unload Me
    MaintenanceForm.Visible = True
End Sub

```

```

Private Sub Command2_Click()
Dim Response As Integer
    'this may produce an error if you delete the last
    'record or the only record in the recordset
    Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
        "Are you sure you want to do this?", vbYesNo)
    If Response = vbYes Then
        If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
            Data1.Recordset.Delete
            Data1.Recordset.MoveNext
            If Data1.Recordset.EOF = True Then
                Data1.Recordset.MovePrevious
            End If
        Else
            MsgBox "There are no records to delete."
        End If
    End If
End Sub

```

```

Private Sub Command3_Click()
    Data1.UpdateRecord
    Data1.Recordset.Bookmark = Data1.Recordset.LastModified
    Beep
End Sub

```

```

Private Sub Command4_Click()
    Data1.Recordset.AddNew
End Sub

```

```

Private Sub cmdClose_Click()
    DolphinViewChangeForm.cmdUpdate.Enabled = True
    MaintenanceForm.Visible = True
    Unload Me
End Sub

```

```

Private Sub cmdDelete_Click()
Dim Response As Integer
    Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
        "Are you sure you want to do this?", vbYesNo)
    If Response = vbYes Then
        If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
            Data1.Recordset.Delete
            Data1.Recordset.MoveNext
            DolphinViewChangeForm.cmdUpdate.Enabled = False
            DolphinViewChangeForm.cmdNew.Enabled = True

```

```

    If Data1.Recordset.EOF = True Then
        Data1.Recordset.MovePrevious
    End If
Else
    MsgBox "There are no records to delete."
End If
End If
End Sub

```

```

Private Sub cmdNew_Click()
Dim i As Integer
If Data1.Recordset.RecordCount = 0 Then
    For i = 0 To 2
        txtFields(i).Enabled = True
    Next
End If
Data1.Recordset.AddNew
DolphinViewChangeForm.cmdUpdate.Enabled = True
DolphinViewChangeForm.cmdDelete.Enabled = False
End Sub

```

```

Private Sub cmdUpdate_Click()
On Error GoTo ErrorHandler
If Left(DolphinViewChangeForm.txtFields(0), 1) <> "" And _
Left(DolphinViewChangeForm.txtFields(0), 1) <> " " And _
Left(DolphinViewChangeForm.txtFields(1), 1) <> "" And _
Left(DolphinViewChangeForm.txtFields(1), 1) <> " " And _
Left(DolphinViewChangeForm.txtFields(2), 1) <> "" And _
Left(DolphinViewChangeForm.txtFields(2), 1) <> " " Then
    Data1.UpdateRecord
    Data1.Recordset.Bookmark = Data1.Recordset.LastModified
    Beep
    DolphinViewChangeForm.cmdNew.Enabled = True
    DolphinViewChangeForm.cmdDelete.Enabled = True
End If
Exit Sub

```

```

ErrorHandler:
MsgBox "Duplicate Record. No new record created."
DolphinViewChangeForm.cmdNew.Enabled = True
DolphinViewChangeForm.cmdDelete.Enabled = True
Data1.Recordset.MoveFirst
End Sub

```

```

Private Sub Data1_Reposition()
DolphinViewChangeForm.cmdUpdate.Enabled = False
DolphinViewChangeForm.cmdNew.Enabled = True
On Error Resume Next
Data1.Caption = "Record: " & (Data1.Recordset.AbsolutePosition + 1)
End Sub

```

```

Private Sub Form_Activate()
Dim i As Integer
VideosViewChangeForm.cmdNew.Enabled = True
If Data1.Recordset.RecordCount = 0 Then
    For i = 0 To 2
        txtFields(i).Enabled = False
    Next
End If

```

```

Next
End If
End Sub

```

```

Private Sub txtFields_Change(Index As Integer)
    DolphinViewChangeForm.cmdUpdate.Enabled = True
    DolphinViewChangeForm.cmdNew.Enabled = False
End Sub

```

**** GRAPHICS.FRM ****

THIS SOURCE CODE DEFINES THE GRAPHICS FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO DISPLAY A GRAPH SHOWING THE RELATIONSHIP BETWEEN BLUBBER, GROWTH, AND WEIGHT.

VERSION 4.00

```
Begin VB.Form GraphicsForm
```

```
Begin VB.Menu mnuFile
```

```
    Caption    = "&File"
```

```
    Begin VB.Menu mnuFileExit
```

```
        Caption    = "E&xit"
```

```
    End
```

```
End
```

```
End
```

```
Attribute VB_Name = "GraphicsForm"
```

```
Attribute VB_Creatable = False
```

```
Attribute VB_Exposed = False
```

```
Option Explicit
```

```
Private Sub BuildXAxis(BeginDate As String)
```

```
    If Right(BeginDate, 2) = "01" Then
```

```
        Exit Sub Default X axis will be loaded.
```

```
    End If
```

```
    If Right(BeginDate, 2) = "02" Then
```

```
        Graph1.AutoInc = 1 Turn on the Auto Increment of ThisSet and ThisPoint, 0 = Off.
```

```
        Graph1.LabelText = "F"
```

```
        Graph1.LabelText = "M"
```

```
        Graph1.LabelText = "A"
```

```
        Graph1.LabelText = "M"
```

```
        Graph1.LabelText = "J"
```

```
        Graph1.LabelText = "J"
```

```
        Graph1.LabelText = "A"
```

```
        Graph1.LabelText = "S"
```

```
        Graph1.LabelText = "O"
```

```
        Graph1.LabelText = "N"
```

```
        Graph1.LabelText = "D"
```

```
        Graph1.LabelText = "J"
```

```
        Graph1.AutoInc = 0 Turn back off the Auto Increment of ThisSet and ThisPoint, 1 = On.
```

```
    Exit Sub
```

```
End If
```

```
    If Right(BeginDate, 2) = "03" Then
```

```
        Graph1.AutoInc = 1 Turn on the Auto Increment of ThisSet and ThisPoint, 0 = Off.
```

```
        Graph1.LabelText = "M"
```

```
        Graph1.LabelText = "A"
```

```

Graph1.LabelText = "M"
Graph1.LabelText = "J"
Graph1.LabelText = "J"
Graph1.LabelText = "A"
Graph1.LabelText = "S"
Graph1.LabelText = "O"
Graph1.LabelText = "N"
Graph1.LabelText = "D"
Graph1.LabelText = "J"
Graph1.LabelText = "F"
Graph1.AutoInc = 0 Turn back off the Auto Increment of ThisSet and ThisPoint, 1 = On.
Exit Sub
End If
THIS GOES ON FOR THE REMAINING 10 MONTHS.
MsgBox "An invalid month was entered. Enter date as YYMM."
End Sub

```

```

Private Sub Combo1_Click()
Graph1.GraphType = 4 3D
If Combo1.Text = "Horizontal" Then
Graph1.YAxisMax = 1000
Graph1.YAxisTicks = 10 Nbr of Ticks
Graph1.GraphStyle = 1 Horizontal
Graph1.LeftTitle = "Months"
Graph1.BottomTitle = "cm./lbs."
ElseIf Combo1.Text = "V. Stacked" Then
Graph1.GraphStyle = 2 Vertical Stacked
Graph1.YAxisMax = 1200
Graph1.YAxisTicks = 12 Nbr of Ticks
Graph1.LeftTitle = "cm./lbs."
Graph1.BottomTitle = "Months"
ElseIf Combo1.Text = "H. Stacked" Then
Graph1.GraphStyle = 3 Horizontal Stacked
Graph1.YAxisMax = 1200
Graph1.YAxisTicks = 12 Nbr of Ticks
Graph1.LeftTitle = "Months"
Graph1.BottomTitle = "cm./lbs."
Else
Graph1.GraphStyle = 0 Default Vertical
Graph1.YAxisMax = 1000
Graph1.YAxisTicks = 10 Nbr of Ticks
Graph1.LeftTitle = "cm./lbs."
Graph1.BottomTitle = "Months"
End If
Graph1.DrawMode = 2 Draw, 5=print, 1=clear
End Sub

```

```

Private Sub Combo2_Click()
Graph1.GraphType = 6
Graph1.LeftTitle = "cm./lbs." Needed because 3D Horizontal may change this.
Graph1.BottomTitle = "Months" Needed because 3D Horizontal may change this.
If Combo2.Text = "Lines" Then
Graph1.YAxisMax = 1000
Graph1.YAxisTicks = 10 Nbr of Ticks
Graph1.GraphStyle = 4 Lines
Else

```

```

Graph1.YAxisMax = 1000
Graph1.YAxisTicks = 10 'Nbr of Ticks
Graph1.GraphStyle = 5 'Lines and Symbols
End If
Graph1.DrawMode = 2 'Draw, 5=print, 1=clear
End Sub

```

```

Private Sub Command1_Click()
Dim Message As String * 40
Dim Title As String * 25
Dim Default As String * 4
Dim BeginDate As String * 4
Dim i As Integer
Dim j As Integer
Dim TempID As String * 10
TempID = DBCombo1.Text
Message = "Enter Beginning Year and Month (YYMM)"
Title = "Beginning Search Date" 'Set title.
Default = "9701" 'Set default.
If Data2.Recordset.RecordCount <> 0 Then
Data2.Recordset.MoveFirst
Else
MsgBox "No Records in Monthly Table"
Exit Sub
End If
On Error GoTo ErrorHandler
'Display dialog box at position 1600, 800.
BeginDate = InputBox(Message, Title, Default, 1600, 800)
If Trim(BeginDate) = "" Then 'Cancel button pressed
Exit Sub
End If
Label1 = "Date: " & BeginDate
Do While Data2.Recordset.EOF = False
If (Data2.Recordset.Fields("ID_Number") = DBCombo1.Text) And _
(Data2.Recordset.Fields("Year_Month") = BeginDate) Then
For j = 1 To 12
Graph1.ThisPoint = j
For i = 1 To 3
Graph1.ThisSet = i
If Graph1.ThisSet = 1 Then 'Blubber'
Graph1.GraphData = Data2.Recordset.Fields("Blubber")
ElseIf Graph1.ThisSet = 2 Then
Graph1.GraphData = Data2.Recordset.Fields("Weight")
Else
Graph1.GraphData = Data2.Recordset.Fields("Growth")
End If
Next
If Trim(TempID) <> Trim(Data2.Recordset.Fields("ID_Number")) Then
MsgBox "Data Records have overflowed into " & Data2.Recordset.Fields("ID_Number") & "
records"
Exit Sub
End If
Data2.Recordset.MoveNext
Next
'Graph1.SeeThru = 1 'Turn SeeThru On (Not implemented in VB 4.0) :^(
Call BuildXAxis(BeginDate)

```

```

    Graph1.DrawMode = 2 'Draw, 5=print, 1=clear
    Exit Sub
End If
Data2.Recordset.MoveNext
Loop
MsgBox "Beginning record not located." & Chr(13) & Chr(10) & _
    "Be sure to enter date as YYMM."
Exit Sub
ErrorHandler:
    MsgBox "Record dates go beyond records available"
End Sub

Private Sub Command3_Click()
    Graph1.DataReset = 9 'All Data
    Graph1.DrawMode = 2 'Draw
End Sub

Private Sub Command4_Click()
    Graph1.GraphType = 4 '3D
    If Combo1.Text = "Horizontal" Then
        Graph1.GraphStyle = 1 'Horizontal
        Graph1.YAxisMax = 10
    ElseIf Combo1.Text = "V. Stacked" Then
        Graph1.GraphStyle = 2 'Vertical Stacked
        Graph1.YAxisMax = 30
    ElseIf Combo1.Text = "H. Stacked" Then
        Graph1.GraphStyle = 3 'Horizontal Stacked
        Graph1.YAxisMax = 30
    Else
        Graph1.GraphStyle = 0 'Default Vertical
        Graph1.YAxisMax = 10
    End If
    Graph1.DrawMode = 2 'Draw, 5=print, 1=clear
End Sub

Private Sub Command5_Click()
    Graph1.GraphStyle = 5 'Lines and Symbols
    Graph1.DrawMode = 2 'Draw, 5=print, 1=clear
End Sub

Private Sub Combo3_Click()
    Graph1.GraphType = 8
    Graph1.LeftTitle = "cm./lbs." 'Needed because 3D Horizontal may change this.
    Graph1.BottomTitle = "Months" 'Needed because 3D Horizontal may change this.
    If Combo3.Text = "Area" Then
        Graph1.YAxisMax = 1200
        Graph1.YAxisTicks = 12 'Nbr of Ticks
        Graph1.GraphStyle = 0 'Default Area Graph
    Else
        Graph1.YAxisMax = 1000
        Graph1.YAxisTicks = 10 'Nbr of Ticks
        Graph1.GraphStyle = 1 'Absolute Graph
    End If
    Graph1.DrawMode = 2 'Draw, 5=print, 1=clear
End Sub

```

```

Private Sub Form_Load()
    Combo1.AddItem "Vertical"
    Combo1.AddItem "Horizontal"
    Combo1.AddItem "V. Stacked"
    Combo1.AddItem "H. Stacked"
    Combo2.AddItem "Lines"
    Combo2.AddItem "Lines and Symbols"
    Combo3.AddItem "Area"
    Combo3.AddItem "Absolute"
    Graph1.Background = 15 'White
    Graph1.BorderStyle = 1 'Fixed-Single
    Graph1.BottomTitle = "Months"
    Graph1.DrawStyle = 1 'Color
    Graph1.Foreground = 1 'Blue
    Graph1.GraphTitle = "Blubber/Growth/Weight"
    Graph1.GraphType = 6 'Line, 0=Clear
    Graph1.GraphStyle = 4 'Lines
    Graph1.GridStyle = 1 'Horizontal
    Graph1.LabelEvery = 1 'Label each X-axis mark
    Graph1.LeftTitle = "cm./lbs."
    Graph1.NumSets = 3 'Nbr of lines across graph
    Graph1.NumPoints = 12 'Nbr of marks on X-axis
    Graph1.PrintStyle = 0 'Monochrome
    Graph1.RandomData = 0 'Off
    Graph1.ThickLines = 1 'On
    Graph1.ThisPoint = 1 'Specifies which point
    Graph1.ThisSet = 1 'Specifies which line
    Graph1.TickEvery = 1 'Make a tick at each pt. on X-axis
    Graph1.YAxisMax = 1000 'Upper Value
    Graph1.YAxisMin = 0 'Lower Value
    Graph1.YAxisStyle = 2 'User Defined
    Graph1.YAxisTicks = 10 'Nbr of Ticks

```

This section sets the X-axis labels and the legend. AutoInc needs to be on.

```

Graph1.AutoInc = 1 'Turn on the Auto Increment of ThisSet and ThisPoint, 0 = Off.
Graph1.LabelText = "J"
Graph1.LabelText = "F"
Graph1.LabelText = "M"
Graph1.LabelText = "A"
Graph1.LabelText = "M"
Graph1.LabelText = "J"
Graph1.LabelText = "J"
Graph1.LabelText = "A"
Graph1.LabelText = "S"
Graph1.LabelText = "O"
Graph1.LabelText = "N"
Graph1.LabelText = "D"
Graph1.LegendText = "Blubber"
Graph1.LegendText = "Weight"
Graph1.LegendText = "Growth"
Graph1.AutoInc = 0 'Turn back off the Auto Increment of ThisSet and ThisPoint, 1 = On.
End Sub

```

```

Private Sub Image3_Click()
    Graph1.DrawMode = 5
    MousePointer = vbDefault

```

End Sub

```
Private Sub Image3_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    MousePointer = vbHourglass
End Sub
```

```
Private Sub mnuFileExit_Click()
    Unload Me
End Sub
```

**** LEARN.FRM ****

THIS SOURCE CODE DEFINES THE LEARNED BEHAVIORS FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO ACCESS THE TABLE NAMED *LEARNED_BEHAVIOR*.

VERSION 4.00

```
Begin VB.Form LearnedBehaviorViewChangeForm
Attribute VB_Name = "LearnedBehaviorViewChangeForm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit
```

```
Private Sub cmdAdd_Click()
    Data1.Recordset.AddNew
End Sub
```

```
Private Sub cmdDelete_Click()
Dim Response As Integer
    Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
        "Are you sure you want to do this?", vbYesNo)
    If Response = vbYes Then
        If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
            Data1.Recordset.Delete
            Data1.Recordset.MoveNext
            LearnedBehaviorViewChangeForm.cmdUpdate.Enabled = False
            LearnedBehaviorViewChangeForm.cmdNew.Enabled = True
            If Data1.Recordset.EOF = True Then
                Data1.Recordset.MovePrevious
            End If
        Else
            MsgBox "There are no records to delete."
        End If
    End If
End Sub
```

```
Private Sub cmdNew_Click()
Dim i As Integer
    If Data1.Recordset.RecordCount = 0 Then
        For i = 0 To 2
            txtFields(i).Enabled = True
        Next
    End If
    Data1.Recordset.AddNew
```

```

LearnedBehaviorViewChangeForm.cmdUpdate.Enabled = True
LearnedBehaviorViewChangeForm.cmdDelete.Enabled = False
End Sub

```

```

Private Sub cmdUpdate_Click()
    On Error GoTo ErrorHandler
    If Left(LearnedBehaviorViewChangeForm.txtFields(0), 1) <> "" And _
        Left(LearnedBehaviorViewChangeForm.txtFields(0), 1) <> " " And _
        Left(LearnedBehaviorViewChangeForm.txtFields(1), 1) <> "" And _
        Left(LearnedBehaviorViewChangeForm.txtFields(1), 1) <> " " And _
        Left(LearnedBehaviorViewChangeForm.txtFields(2), 1) <> "" And _
        Left(LearnedBehaviorViewChangeForm.txtFields(2), 1) <> " " Then
        Data1.UpdateRecord
        Data1.Recordset.Bookmark = Data1.Recordset.LastModified
        Beep
        LearnedBehaviorViewChangeForm.cmdNew.Enabled = True
        LearnedBehaviorViewChangeForm.cmdDelete.Enabled = True
    End If
    Exit Sub

```

```

ErrorHandler:
    MsgBox "Duplicate Record. No new record created."
    LearnedBehaviorViewChangeForm.cmdNew.Enabled = True
    LearnedBehaviorViewChangeForm.cmdDelete.Enabled = True
    Data1.Recordset.MoveFirst
End Sub

```

```

Private Sub cmdClose_Click()
    LearnedBehaviorViewChangeForm.cmdUpdate.Enabled = True
    MaintenanceForm.Visible = True
    Unload Me
End Sub

```

```

Private Sub Command1_Click()
    Unload Me
    MaintenanceForm.Visible = True
End Sub

```

```

Private Sub Command2_Click()
    Dim Response As Integer
    'this may produce an error if you delete the last
    'record or the only record in the recordset
    Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
        "Are you sure you want to do this?", vbYesNo)
    If Response = vbYes Then
        If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
            Data1.Recordset.Delete
            Data1.Recordset.MoveNext
            If Data1.Recordset.EOF = True Then
                Data1.Recordset.MovePrevious
            End If
        Else
            MsgBox "There are no records to delete."
        End If
    End If
End Sub

```

```

Private Sub Command3_Click()
    Data1.Recordset.AddNew
End Sub

Private Sub Command4_Click()
    Data1.UpdateRecord
    Data1.Recordset.Bookmark = Data1.Recordset.LastModified
    Beep
End Sub

Private Sub Data1_Reposition()
    LearnedBehaviorViewChangeForm.cmdUpdate.Enabled = False
    LearnedBehaviorViewChangeForm.cmdNew.Enabled = True
    On Error Resume Next
    Data1.Caption = "Record: " & (Data1.Recordset.AbsolutePosition + 1)
End Sub

Private Sub Form_Activate()
    Dim i As Integer
    LearnedBehaviorViewChangeForm.cmdNew.Enabled = True
    If Data1.Recordset.RecordCount = 0 Then
        For i = 0 To 2
            txtFields(i).Enabled = False
        Next
    End If
End Sub

Private Sub txtFields_Change(Index As Integer)
    LearnedBehaviorViewChangeForm.cmdUpdate.Enabled = True
    LearnedBehaviorViewChangeForm.cmdNew.Enabled = False
End Sub

```

**** MAINTENANCEFORM.FRM ****

THIS SOURCE CODE DEFINES THE MAINTENANCE FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO DISPLAY THE NAMES OF THE INPUT FORMS, TABLES, AND USER-DEFINED TABLES AVAILABLE TO THE USER.

VERSION 4.00

```

Begin VB.Form MaintenanceForm
Attribute VB_Name = "MaintenanceForm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

```

```

Private Sub Daily_Click()
    Load frmDaily
    frmDaily.Visible = True
    MaintenanceForm.Tag = "Triggered by Maintenance Form"
    MaintenanceForm.Visible = False
End Sub

```

```

Private Sub Form_Activate()
    MousePointer = vbDefault

```

End Sub

```
Private Sub Label14_Click()  
    Load frmProject  
    frmProject.Visible = True  
    MaintenanceForm.Visible = False  
End Sub
```

```
Private Sub Label15_Click()  
    Load frmCharact  
    frmCharact.Visible = True  
    MaintenanceForm.Visible = False  
End Sub
```

```
Private Sub Label6_Click()  
    Load frmSessionT  
    frmSessionT.Visible = True  
    MaintenanceForm.Visible = False  
End Sub
```

```
Private Sub Command1_Click()  
    Unload Me  
    ProgenyMainForm.Visible = True  
End Sub
```

```
Private Sub Label1_Click()  
    Load frmSubTypes  
    frmSubTypes.Visible = True  
    MaintenanceForm.Visible = False  
End Sub
```

```
Private Sub Label10_Click()  
    Load SessionsViewChangeForm  
    SessionsViewChangeForm.Visible = True  
    MaintenanceForm.Tag = "Triggered by Maintenance Form"  
    MaintenanceForm.Visible = False  
End Sub
```

```
Private Sub Label11_Click()  
    Load LearnedBehaviorViewChangeForm  
    LearnedBehaviorViewChangeForm.Visible = True  
    MaintenanceForm.Visible = False  
End Sub
```

```
Private Sub Label12_Click()  
    Load TypicalViewChangeForm  
    TypicalViewChangeForm.Visible = True  
    MaintenanceForm.Tag = "Triggered by Maintenance Form"  
    MaintenanceForm.Visible = False  
End Sub
```

```
Private Sub Label13_Click()  
    Load VideosViewChangeForm  
    VideosViewChangeForm.Visible = True  
    MaintenanceForm.Visible = False  
End Sub
```

```
Private Sub Label3_Click()
    Load DolphinViewChangeForm
    DolphinViewChangeForm.Visible = True
    MaintenanceForm.Visible = False
End Sub
```

```
Private Sub Label4_Click()
    MousePointer = vbHourglass
    Load DailyInputForm
    Load DailyInputContForm
    DailyInputForm.Visible = True
    MaintenanceForm.Visible = False
End Sub
```

```
Private Sub Label5_Click()
    MousePointer = vbHourglass
    Load MonthlyInputForm
    MonthlyInputForm.Visible = True
    MaintenanceForm.Visible = False
End Sub
```

```
Private Sub Label8_Click()
    Load frmConstants
    frmConstants.Visible = True
    MaintenanceForm.Tag = "Triggered by Maintenance Form"
    MaintenanceForm.Visible = False
End Sub
```

```
Private Sub Label9_Click()
    Load TrainersViewChangeForm
    TrainersViewChangeForm.Visible = True
    MaintenanceForm.Visible = False
End Sub
```

**** MONTHLYINPUTFORM.FRM ****

THIS SOURCE CODE DEFINES THE MONTHLY INPUT FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO ACCESS THE TABLE NAMED MONTHLY_STAT.

```
Begin VB.Form MonthlyInputForm
Attribute VB_Name = "MonthlyInputForm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit
```

```
Private Sub cmdAdd_Click()
    Data1.Recordset.AddNew
End Sub
```

```
Private Sub cmdDelete_Click()
Dim Response As Integer
    Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
        "Are you sure you want to do this?", vbYesNo)
```

```

If Response = vbYes Then
    If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
        Data1.Recordset.Delete
        Data1.Recordset.MoveNext
        MonthlyInputForm.cmdUpdate.Enabled = False
        MonthlyInputForm.cmdNew.Enabled = True
        If Data1.Recordset.EOF = True Then
            Data1.Recordset.MovePrevious
        End If
    Else
        MsgBox "There are no records to delete."
    End If
End If
End Sub

```

```

Private Sub cmdNew_Click()
Dim i As Integer
    If Data1.Recordset.RecordCount = 0 Then
        For i = 0 To 4
            txtFields(i).Enabled = True
        Next
    End If
    Data1.Recordset.AddNew
    MonthlyInputForm.cmdUpdate.Enabled = True
    MonthlyInputForm.cmdDelete.Enabled = False
End Sub

```

```

Private Sub cmdUpdate_Click()
    On Error GoTo ErrorHandler
    If Left(MonthlyInputForm.txtFields(0), 1) <> "" And _
        Left(MonthlyInputForm.txtFields(0), 1) <> " " And _
        Left(MonthlyInputForm.txtFields(1), 1) <> "" And _
        Left(MonthlyInputForm.txtFields(1), 1) <> " " And _
        Left(MonthlyInputForm.txtFields(2), 1) <> "" And _
        Left(MonthlyInputForm.txtFields(2), 1) <> " " And _
        Left(MonthlyInputForm.txtFields(3), 1) <> "" And _
        Left(MonthlyInputForm.txtFields(3), 1) <> " " And _
        Left(MonthlyInputForm.txtFields(4), 1) <> "" And _
        Left(MonthlyInputForm.txtFields(4), 1) <> " " Then
        Data1.UpdateRecord
        Data1.Recordset.Bookmark = Data1.Recordset.LastModified
        Beep
        MonthlyInputForm.cmdNew.Enabled = True
        MonthlyInputForm.cmdDelete.Enabled = True
    End If
    Exit Sub
ErrorHandler:
    MsgBox "Duplicate Record. No new record created."
    MonthlyInputForm.cmdNew.Enabled = True
    MonthlyInputForm.cmdDelete.Enabled = True
    Data1.Recordset.MoveFirst
End Sub

```

```

Private Sub cmdClose_Click()
    MonthlyInputForm.cmdUpdate.Enabled = True
    MaintenanceForm.Visible = True

```

```
Unload Me
End Sub
```

```
Private Sub Data1_Reposition()
    MonthlyInputForm.cmdUpdate.Enabled = False
    MonthlyInputForm.cmdNew.Enabled = True
    On Error Resume Next
    Data1.Caption = "Record: " & (Data1.Recordset.AbsolutePosition + 1)
End Sub
```

```
Private Sub Form_Activate()
    Dim i As Integer
    MonthlyInputForm.cmdNew.Enabled = True
    If Data1.Recordset.RecordCount = 0 Then
        For i = 0 To 4
            txtFields(i).Enabled = False
        Next
    End If
End Sub
```

```
Private Sub txtFields_Change(Index As Integer)
    MonthlyInputForm.cmdUpdate.Enabled = True
    MonthlyInputForm.cmdNew.Enabled = False
End Sub
```

**** MONTHLYSUMMARY.FRM ****

THIS SOURCE CODE DEFINES THE MONTHLY GRAPHICS FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO DISPLAY A GRAPH SHOWING THE RELATIONSHIP BETWEEN FOOD INTAKE AND RATING.

```
Begin VB.Form frmMonthly
Begin VB.Menu mnuFile
    Caption      = "&File"
    Begin VB.Menu mnuFileExit
        Caption    = "E&xit"
    End
End
Attribute VB_Name = "frmMonthly"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit
```

```
Private Sub Combo1_Click()
    grfMonthly.GraphType = 4 '3D
    If Combo1.Text = "Horizontal" Then
        grfMonthly.YAxisMax = 30
        grfMonthly.YAxisTicks = 30 'Nbr of Ticks
        grfMonthly.GraphStyle = 1 'Horizontal
        grfMonthly.LeftTitle = "Day of Month"
        grfMonthly.BottomTitle = ""
    ElseIf Combo1.Text = "V. Stacked" Then
        grfMonthly.GraphStyle = 2 'Vertical Stacked
    End If
End Sub
```

```

    grfMonthly.YAxisMax = 35
    grfMonthly.YAxisTicks = 35 Nbr of Ticks
    grfMonthly.LeftTitle = ""
    grfMonthly.BottomTitle = "Day of Month"
ElseIf Combo1.Text = "H. Stacked" Then
    grfMonthly.GraphStyle = 3 Horizontal Stacked
    grfMonthly.YAxisMax = 35
    grfMonthly.YAxisTicks = 35 Nbr of Ticks
    grfMonthly.LeftTitle = "Day of Month"
    grfMonthly.BottomTitle = ""
Else
    grfMonthly.GraphStyle = 0 Default Vertical
    grfMonthly.YAxisMax = 30
    grfMonthly.YAxisTicks = 30 Nbr of Ticks
    grfMonthly.LeftTitle = ""
    grfMonthly.BottomTitle = "Day of Month"
End If
grfMonthly.DrawMode = 2 Draw, 5=print, 1=clear
End Sub

Private Sub Combo2_Click()
    grfMonthly.GraphType = 6
    grfMonthly.LeftTitle = "" Needed because 3D Horizontal may change this.
    grfMonthly.BottomTitle = "Day of Month" Needed because 3D Horizontal may change this.
    If Combo2.Text = "Lines" Then
        grfMonthly.YAxisMax = 30
        grfMonthly.YAxisTicks = 30 Nbr of Ticks
        grfMonthly.GraphStyle = 4 Lines
    Else
        grfMonthly.YAxisMax = 30
        grfMonthly.YAxisTicks = 30 Nbr of Ticks
        grfMonthly.GraphStyle = 5 Lines and Symbols
    End If
    grfMonthly.DrawMode = 2 Draw, 5=print, 1=clear
End Sub

Private Sub Combo3_Click()
    grfMonthly.GraphType = 8
    grfMonthly.LeftTitle = "" Needed because 3D Horizontal may change this.
    grfMonthly.BottomTitle = "Day of Month" Needed because 3D Horizontal may change this.
    If Combo3.Text = "Area" Then
        grfMonthly.YAxisMax = 35
        grfMonthly.YAxisTicks = 35 Nbr of Ticks
        grfMonthly.GraphStyle = 0 Default Area Graph
    Else
        grfMonthly.YAxisMax = 30
        grfMonthly.YAxisTicks = 30 Nbr of Ticks
        grfMonthly.GraphStyle = 1 Absolute Graph
    End If
    grfMonthly.DrawMode = 2 Draw, 5=print, 1=clear
End Sub

Private Sub Command1_Click()
    Dim Message As String * 40
    Dim Title As String * 25
    Dim Default As String * 8

```

```

Dim BeginDate As String * 8
Dim i As Integer
Dim j As Integer
Dim TempID As String * 10
TempID = DBCombo1.Text
Message = "Enter as MM/DD/YY, Don't include zeros"
Message = "Enter as MM/DD/YY" & Chr(13) & Chr(10) & "No leading Zeros."
Title = "Beginning Search Date" ' Set title.
Default = "1/1/90" ' Set default.
If Data2.Recordset.RecordCount <> 0 Then
    Data2.Recordset.MoveFirst
Else
    MsgBox "No Records in Daily Statistics table"
    Exit Sub
End If
On Error GoTo ErrorHandler
Display dialog box at position 1600, 800.
BeginDate = InputBox(Message, Title, Default, 1600, 800)
If Trim(BeginDate) = "" Then 'Cancel button pressed
    Exit Sub
End If
Label1 = "Date: " & BeginDate
Do While Data2.Recordset.EOF = False
    If (Data2.Recordset.Fields("ID_Number") = DBCombo1.Text) And _
        (Trim(Data2.Recordset.Fields("Date")) = Trim(BeginDate)) Then
        For j = 1 To 31
            grfMonthly.ThisPoint = j
            For i = 1 To 2
                grfMonthly.ThisSet = i
                If grfMonthly.ThisSet = 1 Then 'Food Intake'
                    grfMonthly.GraphData = Data2.Recordset.Fields("Food_Intake")
                ElseIf grfMonthly.ThisSet = 2 Then 'Rating
                    grfMonthly.GraphData = Data2.Recordset.Fields("Rating_Average")
                End If
            Next
            If Trim(TempID) <> Trim(Data2.Recordset.Fields("ID_Number")) Then
                MsgBox "Data Records have overflowed into " & Data2.Recordset.Fields("ID_Number") & "
records"
                Exit Sub
            End If
            Data2.Recordset.MoveNext
        Next
        'grfMonthly.SeeThru = 1 Turn SeeThru On (Not implemented in VB 4.0) :^(
        grfMonthly.DrawMode = 2 '2=Draw, 5=print, 1=clear
        Exit Sub
    End If
    Data2.Recordset.MoveNext
Loop
MsgBox "Beginning record not located." & Chr(13) & Chr(10) & _
    "Be sure to enter date as MMDDYY."
Exit Sub
ErrorHandler:
MsgBox "Record dates go beyond records available"
End Sub

Private Sub Form_Load()

```

```

Combo1.AddItem "Vertical"
Combo1.AddItem "Horizontal"
Combo1.AddItem "V. Stacked"
Combo1.AddItem "H. Stacked"
Combo2.AddItem "Lines"
Combo2.AddItem "Lines and Symbols"
Combo3.AddItem "Area"
Combo3.AddItem "Absolute"
grfMonthly.GraphType = 6 Line, 0=Clear
grfMonthly.GraphStyle = 4 Lines
grfMonthly.YAxisMax = 30 Upper Value
grfMonthly.YAxisMin = 0 Lower Value
grfMonthly.YAxisStyle = 2 User Defined
grfMonthly.YAxisTicks = 30 Nbr of Ticks
grfMonthly.AutoInc = 1 Turn on the Auto Increment of ThisSet and ThisPoint, 0 = Off.
grfMonthly.LegendText = "Food Intake"
grfMonthly.LegendText = "Rating"
grfMonthly.AutoInc = 0 Turn back off the Auto Increment of ThisSet and ThisPoint, 1 = On.
grfMonthly.AutoInc = 1 Turn on the Auto Increment of ThisSet and ThisPoint, 0 = Off.
grfMonthly.LabelText = "1"
grfMonthly.LabelText = "2"
grfMonthly.LabelText = "3"
CONTINUE TO LABEL ALL 31 TICKS.
grfMonthly.AutoInc = 0 Turn back off the Auto Increment of ThisSet and ThisPoint, 1 = On.
End Sub

Private Sub Image3_Click()
    grfMonthly.DrawMode = 5
    MousePointer = vbDefault
End Sub

Private Sub Image3_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    MousePointer = vbHourglass
End Sub

Private Sub mnuFileExit_Click()
    Unload Me
End Sub

```

**** OUTPUTFORM.FRM ****

THIS SOURCE CODE DEFINES THE OUTPUT FORM AND THE OBJECTS CONTAINED ON THE FORM. THE OUTPUT FORM IS USED TO DISPLAY THE NAMES OF ALL REPORTS AND GRAPHS AVAILABLE TO THE USER.

```

Begin VB.Form OutputForm
Attribute VB_Name = "OutputForm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Private Sub Command1_Click()
    Unload Me
    ProgenyMainForm.Visible = True

```

End Sub

```
Private Sub Label10_Click()  
    CrystalReport1.ReportFileName = "C:\vb\progeny\typical.rpt"  
    CrystalReport1.Action = 1  
End Sub
```

```
Private Sub Label11_Click()  
    CrystalReport1.ReportFileName = "C:\vb\progeny\constants.rpt"  
    CrystalReport1.Action = 1  
End Sub
```

```
Private Sub Label12_Click()  
    CrystalReport1.ReportFileName = "C:\vb\progeny\daily.rpt"  
    CrystalReport1.Action = 1  
End Sub
```

```
Private Sub Label13_Click()  
    CrystalReport1.ReportFileName = "C:\vb\progeny\charact.rpt"  
    CrystalReport1.Action = 1  
End Sub
```

```
Private Sub Label14_Click()  
    CrystalReport1.ReportFileName = "C:\vb\progeny\subtypes.rpt"  
    CrystalReport1.Action = 1  
End Sub
```

```
Private Sub Label17_Click()  
    Load frmMonthly  
    frmMonthly.Visible = True  
End Sub
```

```
Private Sub Label7_Click()  
    Load GraphicsForm  
    GraphicsForm.Visible = True  
End Sub
```

```
Private Sub Label8_Click()  
    CrystalReport1.ReportFileName = "C:\vb\progeny\projects.rpt"  
    CrystalReport1.Action = 1  
End Sub
```

```
Private Sub Label1_Click()  
    CrystalReport1.ReportFileName = "C:\vb\progeny\behaviors.rpt"  
    CrystalReport1.Action = 1  
End Sub
```

```
Private Sub Label2_Click()  
    CrystalReport1.ReportFileName = "C:\vb\progeny\monthly.rpt"  
    CrystalReport1.Action = 1  
End Sub
```

```
Private Sub Label3_Click()  
    CrystalReport1.ReportFileName = "C:\vb\progeny\trainers.rpt"  
    CrystalReport1.Action = 1  
End Sub
```

```
Private Sub Label4_Click()
    CrystalReport1.ReportFileName = "C:\vb\progeny\dolphins.rpt"
    CrystalReport1.Action = 1
End Sub
```

```
Private Sub Label5_Click()
    CrystalReport1.ReportFileName = "C:\vb\progeny\videos.rpt"
    CrystalReport1.Action = 1
End Sub
```

```
Private Sub Label6_Click()
    CrystalReport1.ReportFileName = "C:\vb\progeny\sessions.rpt"
    CrystalReport1.Action = 1
End Sub
```

```
Private Sub Label9_Click()
    CrystalReport1.ReportFileName = "C:\vb\progeny\sesstypes.rpt"
    CrystalReport1.Action = 1
End Sub
```

**** PROGENY FORM.FRM ****

THIS SOURCE CODE BUILDS THE MAIN MENU FOR THE DODDS APPLICATION.

```
Begin VB.Form ProgenyMainForm
Attribute VB_Name = "ProgenyMainForm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit
```

```
Private Sub Label3_Click()
    Load VideosTakenForm
    VideosTakenForm.Visible = True
    ProgenyMainForm.Visible = False
End Sub
```

```
Private Sub Label5_Click()
    Load MaintenanceForm
    MaintenanceForm.Visible = True
    ProgenyMainForm.Visible = False
End Sub
```

```
Private Sub Label9_Click()
    Load OutputForm
    OutputForm.Visible = True
    ProgenyMainForm.Visible = False
End Sub
```

```
Private Sub mnuFileExit_Click()
    Unload Me
End
End Sub
```

```

Private Sub mnuHelpAbout_Click()
    MsgBox "DODDS Program" & Chr(13) & Chr(10) & Chr(10) & "Developed By:" _
    & Chr(13) & Chr(10) & _
    "Randy Brill" & Chr(13) & Chr(10) & _
    "Mark Xitco" & Chr(13) & Chr(10) & _
    "Scott Klappenback" & Chr(13) & Chr(10) & _
    "Joy Ross" & Chr(13) & Chr(10) & _
    "Michael Dodds" & Chr(13) & Chr(10) & Chr(10) & _
    "Written By:" & Chr(13) & Chr(10) & _
    "Michael Dodds"
End Sub

```

**** PROJECT.FRM ****

THIS SOURCE CODE BUILDS THE PROJECTS FORM. IT IS USED TO ACCESS THE TABLE NAMED PROJECT.

```

Begin VB.Form frmProject
Attribute VB_Name = "frmProject"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

```

```

Private Sub cmdAdd_Click()
    Data1.Recordset.AddNew
End Sub

```

```

Private Sub cmdDelete_Click()
Dim Response As Integer
    Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
    "Are you sure you want to do this?", vbYesNo)
    If Response = vbYes Then
        If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
            Data1.Recordset.Delete
            Data1.Recordset.MoveNext
            frmProject.cmdUpdate.Enabled = False
            frmProject.cmdNew.Enabled = True
            If Data1.Recordset.EOF = True Then
                Data1.Recordset.MovePrevious
            End If
        Else
            MsgBox "There are no records to delete."
        End If
    End If
End Sub

```

```

Private Sub cmdNew_Click()
Dim i As Integer
    If Data1.Recordset.RecordCount = 0 Then
        For i = 0 To 3
            txtFields(i).Enabled = True
        Next
    End If
    Data1.Recordset.AddNew

```

```

frmProject.cmdUpdate.Enabled = True
frmProject.cmdDelete.Enabled = False
End Sub

```

```

Private Sub cmdUpdate_Click()
    On Error GoTo ErrorHandler
    If Left(frmProject.txtFields(0), 1) <> "" And _
        Left(frmProject.txtFields(0), 1) <> " " And _
        Left(frmProject.txtFields(1), 1) <> "" And _
        Left(frmProject.txtFields(1), 1) <> " " And _
        Left(frmProject.txtFields(2), 1) <> "" And _
        Left(frmProject.txtFields(2), 1) <> " " And _
        Left(frmProject.txtFields(3), 1) <> "" And _
        Left(frmProject.txtFields(3), 1) <> " " Then
        Data1.UpdateRecord
        Data1.Recordset.Bookmark = Data1.Recordset.LastModified
        Beep
        frmProject.cmdNew.Enabled = True
        frmProject.cmdDelete.Enabled = True
    End If
    Exit Sub

```

```

ErrorHandler:
    MsgBox "Duplicate Record. No new record created."
    frmProject.cmdNew.Enabled = True
    frmProject.cmdDelete.Enabled = True
    Data1.Recordset.MoveFirst
End Sub

```

```

Private Sub cmdClose_Click()
    VideosViewChangeForm.cmdUpdate.Enabled = True
    MaintenanceForm.Visible = True
    Unload Me
End Sub

```

```

Private Sub Data1_Reposition()
    frmProject.cmdUpdate.Enabled = False
    frmProject.cmdNew.Enabled = True
    On Error Resume Next
    Data1.Caption = "Record: " & (Data1.Recordset.AbsolutePosition + 1)
End Sub

```

```

Private Sub Form_Activate()
    Dim i As Integer
    frmProject.cmdNew.Enabled = True
    If Data1.Recordset.RecordCount = 0 Then
        For i = 0 To 3
            txtFields(i).Enabled = False
        Next
    End If
End Sub

```

```

Private Sub txtFields_Change(Index As Integer)
    frmProject.cmdUpdate.Enabled = True
    frmProject.cmdNew.Enabled = False
End Sub

```

**** SESSION#CHARFORM.FRM ****

THIS SOURCE CODE DEFINES A SESSION'S CHARACTERISTICS FORM. IT DEFINES 10 DATABOUND COMBINATION (DBCOMBO) OBJECTS CONTAINED ON THE FORM. THE FORM IS USED TO CAPTURE UP TO TEN CHARACTERISTICS DISPLAYED DURING A SESSION. THE CODING IS IDENTICAL FOR ALL 20 SESSION#CHARFORMS DEFINED IN THE PROGRAM EXCEPT THAT THE # SIGN CAN BE THE NUMBER 1 THROUGH 20.

```
Begin VB.Form Session#CharForm
Attribute VB_Name = "Session#CharForm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit
```

```
Private Sub Command1_Click()
Dim i As Integer
Dim TempVar As String * 25
DailyInputForm.Check#.Value = 0
For i = 0 To 9
    If (Left(DBCombo1(i), 1) <> "") And (Left(DBCombo1(i), 1) <> " ") Then
        DailyInputForm.Check#.Value = 1
    End If
Next
Session#CharForm.Visible = False
DailyInputForm.Visible = True
End Sub
```

**** SESSIONRECORDSFORM.FRM**

THIS SOURCE CODE DEFINES THE SESSION RECORDS FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO DISPLAY THE RECORDS FROM THE SESSION TABLE BY SELECTING THE DOLPHIN ID AND THE DATE.

```
Attribute VB_Name = "SessionRecordsForm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit
```

```
Private Sub Command1_Click()
    DailyInputForm.Visible = True
    Unload SessionRecordsForm
End Sub
```

```
Private Sub Form_Activate()
Dim RecordCount As Integer
Dim found As Boolean
If Data1.Recordset.RecordCount <> 0 Then
    Data1.Recordset.MoveFirst
End If
found = False
RecordCount = 0
Print
Print "    ID: " & DailyInputForm.DBCombo17.Text & " " & _
```

```

        "Date: " & DailyInputForm.txtFields(0)
Print
Do While Data1.Recordset.EOF = False
    If (Data1.Recordset.Fields("ID_Number") = DailyInputForm.DBCombo17.Text) And _
        (Data1.Recordset.Fields("Date") = DailyInputForm.txtFields(0)) Then
        Print "    Session Type: " & Data1.Recordset.Fields("Session_Type") & _
            Data1.Recordset.Fields("Session_Type_Number") & " " & _
            "    Rating: " & Data1.Recordset.Fields("Rating") & " " & _
            "    Initials: " & Data1.Recordset.Fields("Trainer_Initials") & " " & _
            "    Time: " & Data1.Recordset.Fields("Time") & " " & _
            "    SubType: " & Data1.Recordset.Fields("SubType")
        found = True
        RecordCount = RecordCount + 1
    End If
    Data1.Recordset.MoveNext
Loop
If found = False Then
    MsgBox "Dolphin's Session Records not located."
End If
Print
Print "    Record Count is: " & RecordCount
End Sub

```

**** SESSIONS.FRM ****

THIS SOURCE CODE DEFINES THE SESSIONS FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO ACCESS THE TABLE NAMED *SESSION*.

```

Begin VB.Form SessionsViewChangeForm
Attribute VB_Name = "SessionsViewChangeForm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Private Sub cmdAdd_Click()
    Data1.Recordset.AddNew
End Sub

Private Sub cmdDelete_Click()
Dim Response As Integer
Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
    "    Are you sure you want to do this?", vbYesNo)
If Response = vbYes Then
    If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
        Data1.Recordset.Delete
        Data1.Recordset.MoveNext
        SessionsViewChangeForm.cmdUpdate.Enabled = False
        SessionsViewChangeForm.cmdNew.Enabled = True
        If Data1.Recordset.EOF = True Then
            Data1.Recordset.MovePrevious
        End If
    Else
        MsgBox "There are no records to delete."
    End If
End Sub

```

```
End If
End Sub
```

```
Private Sub cmdNew_Click()
Dim i As Integer
If Data1.Recordset.RecordCount = 0 Then
    For i = 0 To 7
        txtFields(i).Enabled = True
    Next
End If
Data1.Recordset.AddNew
SessionsViewChangeForm.cmdUpdate.Enabled = True
SessionsViewChangeForm.cmdDelete.Enabled = False
End Sub
```

```
Private Sub cmdUpdate_Click()
On Error GoTo ErrorHandler
If Left(SessionsViewChangeForm.txtFields(0), 1) <> "" And _
Left(SessionsViewChangeForm.txtFields(0), 1) <> " " And _
Left(SessionsViewChangeForm.txtFields(1), 1) <> "" And _
Left(SessionsViewChangeForm.txtFields(1), 1) <> " " And _
Left(SessionsViewChangeForm.txtFields(2), 1) <> "" And _
Left(SessionsViewChangeForm.txtFields(2), 1) <> " " And _
Left(SessionsViewChangeForm.txtFields(3), 1) <> "" And _
Left(SessionsViewChangeForm.txtFields(3), 1) <> " " Then
    Data1.UpdateRecord
    Data1.Recordset.Bookmark = Data1.Recordset.LastModified
    Beep
    SessionsViewChangeForm.cmdNew.Enabled = True
    If MaintenanceForm.Tag = "Triggered by Maintenance Form" Then
        SessionsViewChangeForm.cmdDelete.Enabled = True
    End If
End If
Exit Sub
ErrorHandler:
MsgBox "Duplicate Record. No new record created."
SessionsViewChangeForm.cmdNew.Enabled = True
If MaintenanceForm.Tag = "Triggered by Maintenance Form" Then
    SessionsViewChangeForm.cmdDelete.Enabled = True
End If
Data1.Recordset.MoveFirst
End Sub
```

```
Private Sub cmdClose_Click()
If MaintenanceForm.Tag = "Triggered by Maintenance Form" Then
    MaintenanceForm.Visible = True
    MaintenanceForm.Tag = ""
Else
    DailyInputForm.Visible = True
End If
Unload SessionsViewChangeForm
End Sub
```

```
Private Sub Command1_Click()
If MaintenanceForm.Tag = "Triggered by Maintenance Form" Then
    MaintenanceForm.Visible = True
End Sub
```

```

        MaintenanceForm.Tag = ""
    Else
        DailyInputForm.Visible = True
        SessionsViewChangeForm.cmdDelete.Enabled = True
    End If
    Unload SessionsViewChangeForm
End Sub

Private Sub Command2_Click()
    Dim Response As Integer
    'this may produce an error if you delete the last
    'record or the only record in the recordset
    Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
        "Are you sure you want to do this?", vbYesNo)
    If Response = vbYes Then
        Data1.Recordset.Delete
        Data1.Recordset.MoveNext
    End If
End Sub

Private Sub Command3_Click()
    Data1.Recordset.AddNew
End Sub

Private Sub Command4_Click()
    Data1.UpdateRecord
    Data1.Recordset.Bookmark = Data1.Recordset.LastModified
    Beep
End Sub

Private Sub Command5_Click()
    Dim MyCriteria As String
    Dim DolphinID As String * 10
    Dim SessionDate As String * 8
    DolphinID = Trim(DBCombo1.Text)
    SessionDate = Trim(Text2.Text)
    MyCriteria = "(ID_Number = " & "" & DolphinID & ")" & " AND " & "(CStr(Date) = " & "" & _
    SessionDate & ")"
    SessionsViewChangeForm.Data1.Recordset.FindFirst MyCriteria
    If Data1.Recordset.NoMatch Then
        MsgBox "Record Not Located"
    End If
End Sub

Private Sub Data1_Reposition()
    SessionsViewChangeForm.cmdUpdate.Enabled = False
    SessionsViewChangeForm.cmdNew.Enabled = True
    On Error Resume Next
    Data1.Caption = "Record: " & (Data1.Recordset.AbsolutePosition + 1)
End Sub

Private Sub Form_Activate()
    Dim i As Integer
    SessionsViewChangeForm.cmdNew.Enabled = True
    If Data1.Recordset.RecordCount = 0 Then
        For i = 0 To 7
    
```

```

        txtFields(i).Enabled = False
    Next
End If
End Sub

```

```

Private Sub Form_Load()
Dim today As Date
today = Now
today = Format(today, "mm/dd/yy")
Text2.Text = today
If DailyInputForm.Tag = "From Daily Input Form" Then
    SessionsViewChangeForm.cmdDelete.Enabled = False
    DailyInputForm.Tag = ""
End If
End Sub

```

```

Private Sub txtFields_Change(Index As Integer)
    SessionsViewChangeForm.cmdUpdate.Enabled = True
    SessionsViewChangeForm.cmdNew.Enabled = False
End Sub

```

**** SESSIONT.FRM ****

THIS SOURCE CODE DEFINES THE SESSION TYPES FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO ACCESS THE USER DEFINED TABLE NAMED *SESSION_TYPES*.

```

Begin VB.Form frmSessionT
Attribute VB_Name = "frmSessionT"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

```

```

Private Sub cmdAdd_Click()
    Data1.Recordset.AddNew
End Sub

```

```

Private Sub cmdDelete_Click()
Dim Response As Integer
Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
    "Are you sure you want to do this?", vbYesNo)
If Response = vbYes Then
    If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
        Data1.Recordset.Delete
        Data1.Recordset.MoveNext
        frmSessionT.cmdUpdate.Enabled = False
        frmSessionT.cmdNew.Enabled = True
        If Data1.Recordset.EOF = True Then
            Data1.Recordset.MovePrevious
        End If
    Else
        MsgBox "There are no records to delete."
    End If
End If

```

End Sub

```
Private Sub cmdNew_Click()  
    If Data1.Recordset.RecordCount = 0 Then  
        txtFields(0).Enabled = True  
    End If  
    Data1.Recordset.AddNew  
    frmSessionT.cmdUpdate.Enabled = True  
    frmSessionT.cmdDelete.Enabled = False  
End Sub
```

```
Private Sub cmdUpdate_Click()  
    On Error GoTo ErrorHandler  
    If Left(frmSessionT.txtFields(0), 1) <> "" And _  
        Left(frmSessionT.txtFields(0), 1) <> " " Then  
        Data1.UpdateRecord  
        Data1.Recordset.Bookmark = Data1.Recordset.LastModified  
        Beep  
        frmSessionT.cmdNew.Enabled = True  
        frmSessionT.cmdDelete.Enabled = True  
    End If  
Exit Sub
```

```
ErrorHandler:  
    MsgBox "Duplicate Record. No new record created."  
    frmSessionT.cmdNew.Enabled = True  
    frmSessionT.cmdDelete.Enabled = True  
    Data1.Recordset.MoveFirst  
End Sub
```

```
Private Sub cmdClose_Click()  
    frmSessionT.cmdUpdate.Enabled = True  
    MaintenanceForm.Visible = True  
    Unload Me  
End Sub
```

```
Private Sub Data1_Reposition()  
    frmSessionT.cmdUpdate.Enabled = False  
    frmSessionT.cmdNew.Enabled = True  
    On Error Resume Next  
    Data1.Caption = "Record: " & (Data1.Recordset.AbsolutePosition + 1)  
End Sub
```

```
Private Sub Form_Activate()  
    frmSessionT.cmdNew.Enabled = True  
    If Data1.Recordset.RecordCount = 0 Then  
        txtFields(0).Enabled = False  
    End If  
End Sub
```

```
Private Sub txtFields_Change(Index As Integer)  
    frmSessionT.cmdUpdate.Enabled = True  
    frmSessionT.cmdNew.Enabled = False  
End Sub
```

**** STATUS.FRM ****

THIS SOURCE CODE DEFINES THE STATUS FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO DISPLAY MESSAGES TO THE USER WHERE APPROPRIATE.

```
Begin VB.Form frmStatus
Attribute VB_Name = "frmStatus"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit
```

```
Private Sub Command1_Click()
    Unload frmStatus
End Sub
```

**** SUBTYPES.FRM ****

THIS SOURCE CODE DEFINES THE SUBTYPES FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO ACCESS THE USER DEFINED TABLE NAMED *SUBTYPES*.

```
Begin VB.Form frmSubTypes
Attribute VB_Name = "frmSubTypes"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit
```

```
Private Sub cmdAdd_Click()
    Data1.Recordset.AddNew
End Sub
```

```
Private Sub cmdDelete_Click()
Dim Response As Integer
    Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
        "Are you sure you want to do this?", vbYesNo)
    If Response = vbYes Then
        If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
            Data1.Recordset.Delete
            Data1.Recordset.MoveNext
            frmSubTypes.cmdUpdate.Enabled = False
            frmSubTypes.cmdNew.Enabled = True
            If Data1.Recordset.EOF = True Then
                Data1.Recordset.MovePrevious
            End If
        Else
            MsgBox "There are no records to delete."
        End If
    End If
End Sub
```

```
Private Sub cmdNew_Click()
    If Data1.Recordset.RecordCount = 0 Then
        txtFields(0).Enabled = True
    End If
```

```

Data1.Recordset.AddNew
frmSubTypes.cmdUpdate.Enabled = True
frmSubTypes.cmdDelete.Enabled = False
End Sub

Private Sub cmdUpdate_Click()
On Error GoTo ErrorHandler
If Left(frmSubTypes.txtFields(0), 1) <> "" And _
Left(frmSubTypes.txtFields(0), 1) <> " " Then
Data1.UpdateRecord
Data1.Recordset.Bookmark = Data1.Recordset.LastModified
Beep
frmSubTypes.cmdNew.Enabled = True
frmSubTypes.cmdDelete.Enabled = True
End If
Exit Sub
ErrorHandler:
MsgBox "Duplicate Record. No new record created."
frmSubTypes.cmdNew.Enabled = True
frmSubTypes.cmdDelete.Enabled = True
Data1.Recordset.MoveFirst
End Sub

Private Sub cmdClose_Click()
frmSubTypes.cmdUpdate.Enabled = True
MaintenanceForm.Visible = True
Unload Me
End Sub

Private Sub Data1_Reposition()
frmSubTypes.cmdUpdate.Enabled = False
frmSubTypes.cmdNew.Enabled = True
On Error Resume Next
Data1.Caption = "Record: " & (Data1.Recordset.AbsolutePosition + 1)
End Sub

Private Sub Form_Activate()
frmSubTypes.cmdNew.Enabled = True
If Data1.Recordset.RecordCount = 0 Then
txtFields(0).Enabled = False
End If
End Sub

Private Sub txtFields_Change(Index As Integer)
frmSubTypes.cmdUpdate.Enabled = True
frmSubTypes.cmdNew.Enabled = False
End Sub

```

**** TRAINERS.FRM ****

THIS SOURCE CODE DEFINES THE TRAINERS FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO ACCESS THE TABLE NAMED *TRAINER*.

Begin VB.Form TrainersViewChangeForm

```

Attribute VB_Name = "TrainersViewChangeForm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

```

```

Private Sub cmdAdd_Click()
    Data1.Recordset.AddNew
End Sub

```

```

Private Sub cmdDelete_Click()
Dim Response As Integer
    Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
        "Are you sure you want to do this?", vbYesNo)
    If Response = vbYes Then
        If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
            Data1.Recordset.Delete
            Data1.Recordset.MoveNext
            TrainersViewChangeForm.cmdUpdate.Enabled = False
            TrainersViewChangeForm.cmdNew.Enabled = True
            If Data1.Recordset.EOF = True Then
                Data1.Recordset.MovePrevious
            End If
        Else
            MsgBox "There are no records to delete."
        End If
    End If
End Sub

```

```

Private Sub cmdNew_Click()
Dim i As Integer
    If Data1.Recordset.RecordCount = 0 Then
        For i = 0 To 3
            txtFields(i).Enabled = True
        Next
    End If
    Data1.Recordset.AddNew
    TrainersViewChangeForm.cmdUpdate.Enabled = True
    TrainersViewChangeForm.cmdDelete.Enabled = False
End Sub

```

```

Private Sub cmdUpdate_Click()
    On Error GoTo ErrorHandler
    If Left(TrainersViewChangeForm.txtFields(0), 1) <> "" And _
        Left(TrainersViewChangeForm.txtFields(0), 1) <> " " And _
        Left(TrainersViewChangeForm.txtFields(1), 1) <> "" And _
        Left(TrainersViewChangeForm.txtFields(1), 1) <> " " And _
        Left(TrainersViewChangeForm.txtFields(2), 1) <> "" And _
        Left(TrainersViewChangeForm.txtFields(2), 1) <> " " And _
        Left(TrainersViewChangeForm.txtFields(3), 1) <> "" And _
        Left(TrainersViewChangeForm.txtFields(3), 1) <> " " Then
        Data1.UpdateRecord
        Data1.Recordset.Bookmark = Data1.Recordset.LastModified
        Beep
        TrainersViewChangeForm.cmdNew.Enabled = True
        TrainersViewChangeForm.cmdDelete.Enabled = True
    End If

```

```

Exit Sub
ErrorHandler:
MsgBox "Duplicate Record. No new record created."
TrainersViewChangeForm.cmdNew.Enabled = True
TrainersViewChangeForm.cmdDelete.Enabled = True
Data1.Recordset.MoveFirst
End Sub

Private Sub cmdClose_Click()
TrainersViewChangeForm.cmdUpdate.Enabled = True
MaintenanceForm.Visible = True
Unload Me
End Sub

Private Sub Command1_Click()
Unload Me
MaintenanceForm.Visible = True
End Sub

Private Sub Command2_Click()
Dim Response As Integer
'this may produce an error if you delete the last
'record or the only record in the recordset
Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
"Are you sure you want to do this?", vbYesNo)
If Response = vbYes Then
If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
Data1.Recordset.Delete
Data1.Recordset.MoveNext
If Data1.Recordset.EOF = True Then
Data1.Recordset.MovePrevious
End If
Else
MsgBox "There are no records to delete."
End If
End If
End Sub

Private Sub Command3_Click()
Data1.UpdateRecord
Data1.Recordset.Bookmark = Data1.Recordset.LastModified
Beep
End Sub

Private Sub Command4_Click()
Data1.Recordset.AddNew
End Sub

Private Sub Data1_Reposition()
TrainersViewChangeForm.cmdUpdate.Enabled = False
TrainersViewChangeForm.cmdNew.Enabled = True
On Error Resume Next
Data1.Caption = "Record: " & (Data1.Recordset.AbsolutePosition + 1)
End Sub

Private Sub Form_Activate()

```

```

Dim i As Integer
TrainersViewChangeForm.cmdNew.Enabled = True
If Data1.Recordset.RecordCount = 0 Then
    For i = 0 To 3
        txtFields(i).Enabled = False
    Next
End If
End Sub

Private Sub txtFields_Change(Index As Integer)
    TrainersViewChangeForm.cmdUpdate.Enabled = True
    TrainersViewChangeForm.cmdNew.Enabled = False
End Sub

```

**** TYPICAL.FRM ****

THIS SOURCE CODE DEFINES THE TYPICAL CHARACTERISTICS FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO ACCESS THE TABLE NAMED *TYPICAL_CHARACTERISTICS*.

```

Begin VB.Form TypicalViewChangeForm
Attribute VB_Name = "TypicalViewChangeForm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Private Sub cmdAdd_Click()
    Data1.Recordset.AddNew
End Sub

Private Sub cmdDelete_Click()
Dim Response As Integer
    Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
        "Are you sure you want to do this?", vbYesNo)
    If Response = vbYes Then
        If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
            Data1.Recordset.Delete
            Data1.Recordset.MoveNext
            TypicalViewChangeForm.cmdUpdate.Enabled = False
            TypicalViewChangeForm.cmdNew.Enabled = True
            If Data1.Recordset.EOF = True Then
                Data1.Recordset.MovePrevious
            End If
        Else
            MsgBox "There are no records to delete."
        End If
    End If
End Sub

Private Sub cmdNew_Click()
Dim i As Integer
    If Data1.Recordset.RecordCount = 0 Then
        For i = 0 To 4
            txtFields(i).Enabled = True
        Next
    End If
End Sub

```

```

    Next
End If
Data1.Recordset.AddNew
TypicalViewChangeForm.cmdUpdate.Enabled = True
TypicalViewChangeForm.cmdDelete.Enabled = False
End Sub

Private Sub cmdUpdate_Click()
    On Error GoTo ErrorHandler
    If Left(TypicalViewChangeForm.txtFields(0), 1) <> "" And _
        Left(TypicalViewChangeForm.txtFields(0), 1) <> " " And _
        Left(TypicalViewChangeForm.txtFields(1), 1) <> "" And _
        Left(TypicalViewChangeForm.txtFields(1), 1) <> " " And _
        Left(TypicalViewChangeForm.txtFields(2), 1) <> "" And _
        Left(TypicalViewChangeForm.txtFields(2), 1) <> " " And _
        Left(TypicalViewChangeForm.txtFields(3), 1) <> "" And _
        Left(TypicalViewChangeForm.txtFields(3), 1) <> " " Then
        Data1.UpdateRecord
        Data1.Recordset.Bookmark = Data1.Recordset.LastModified
        Beep
        TypicalViewChangeForm.cmdNew.Enabled = True
        If MaintenanceForm.Tag = "Triggered by Maintenance Form" Then
            TypicalViewChangeForm.cmdDelete.Enabled = True
        End If
    End If
    Exit Sub
ErrorHandler:
    MsgBox "Duplicate Record. No new record created."
    TypicalViewChangeForm.cmdNew.Enabled = True
    If MaintenanceForm.Tag = "Triggered by Maintenance Form" Then
        TypicalViewChangeForm.cmdDelete.Enabled = True
    End If
    Data1.Recordset.MoveFirst
End Sub

Private Sub cmdClose_Click()
    If MaintenanceForm.Tag = "Triggered by Maintenance Form" Then
        MaintenanceForm.Visible = True
        MaintenanceForm.Tag = ""
    Else
        DailyInputForm.Visible = True
    End If
    Unload TypicalViewChangeForm
End Sub

Private Sub Command1_Click()
    If MaintenanceForm.Tag = "Triggered by Maintenance Form" Then
        MaintenanceForm.Visible = True
        MaintenanceForm.Tag = ""
    Else
        DailyInputForm.Visible = True
        TypicalViewChangeForm.cmdDelete.Enabled = True
    End If
    Unload TypicalViewChangeForm
End Sub

```

```

Private Sub Command2_Click()
Dim Response As Integer
'this may produce an error if you delete the last
'record or the only record in the recordset
Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
"Are you sure you want to do this?", vbYesNo)
If Response = vbYes Then
Data1.Recordset.Delete
Data1.Recordset.MoveNext
End If
End Sub

Private Sub Command3_Click()
Data1.Recordset.AddNew
End Sub

Private Sub Command4_Click()
Data1.UpdateRecord
Data1.Recordset.Bookmark = Data1.Recordset.LastModified
Beep
End Sub

Private Sub Command5_Click()
Dim MyCriteria As String
Dim DolphinID As String * 10
Dim SessionDate As String * 8
DolphinID = Trim(DBCombo1.Text)
SessionDate = Trim(Text2.Text)
MyCriteria = "(ID_Number = " & "" & DolphinID & ")" & " AND " & "(CStr(Date) = " & "" &
SessionDate & ")"
TypicalViewChangeForm.Data1.Recordset.FindFirst MyCriteria
If Data1.Recordset.NoMatch Then
MsgBox "Record Not Located"
End If
End Sub

Private Sub Data1_Reposition()
TypicalViewChangeForm.cmdUpdate.Enabled = False
TypicalViewChangeForm.cmdNew.Enabled = True
On Error Resume Next
Data1.Caption = "Record: " & (Data1.Recordset.AbsolutePosition + 1)
End Sub

Private Sub Form_Activate()
Dim i As Integer
VideosViewChangeForm.cmdNew.Enabled = True
If Data1.Recordset.RecordCount = 0 Then
For i = 0 To 4
txtFields(i).Enabled = False
Next
End If
End Sub

Private Sub Form_Load()
Dim today As Date
today = Now

```

```

today = Format(today, "mm/dd/yy")
Text2.Text = today
If DailyInputForm.Tag = "From Daily Input Form" Then
    TypicalViewChangeForm.cmdDelete.Enabled = False
    DailyInputForm.Tag = ""
End If
End Sub
Private Sub txtFields_Change(Index As Integer)
    TypicalViewChangeForm.cmdUpdate.Enabled = True
    TypicalViewChangeForm.cmdNew.Enabled = False
End Sub

```

**** VIDEOS.FRM ****

THIS SOURCE CODE DEFINES THE VIDEOS FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO ACCESS THE TABLE NAMED *VIDEO*.

```

Begin VB.Form VideosViewChangeForm
Attribute VB_Name = "VideosViewChangeForm"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit

Private Sub cmdAdd_Click()
    Data1.Recordset.AddNew
End Sub

Private Sub cmdDelete_Click()
Dim Response As Integer
    Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
        "Are you sure you want to do this?", vbYesNo)
    If Response = vbYes Then
        If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
            Data1.Recordset.Delete
            Data1.Recordset.MoveNext
            VideosViewChangeForm.cmdUpdate.Enabled = False
            VideosViewChangeForm.cmdNew.Enabled = True
            If Data1.Recordset.EOF = True Then
                Data1.Recordset.MovePrevious
            End If
        Else
            MsgBox "There are no records to delete."
        End If
    End If
End Sub

Private Sub cmdNew_Click()
Dim i As Integer
    If Data1.Recordset.RecordCount = 0 Then
        For i = 0 To 3
            txtFields(i).Enabled = True
        Next
    End If
    Data1.Recordset.AddNew

```

```
VideosViewChangeForm.cmdUpdate.Enabled = True
VideosViewChangeForm.cmdDelete.Enabled = False
End Sub
```

```
Private Sub cmdUpdate_Click()
On Error GoTo ErrorHandler
If Left(VideosViewChangeForm.txtFields(0), 1) <> "" And _
Left(VideosViewChangeForm.txtFields(0), 1) <> " " And _
Left(VideosViewChangeForm.txtFields(1), 1) <> "" And _
Left(VideosViewChangeForm.txtFields(1), 1) <> " " And _
Left(VideosViewChangeForm.txtFields(2), 1) <> "" And _
Left(VideosViewChangeForm.txtFields(2), 1) <> " " And _
Left(VideosViewChangeForm.txtFields(3), 1) <> "" And _
Left(VideosViewChangeForm.txtFields(3), 1) <> " " Then
Data1.UpdateRecord
Data1.Recordset.Bookmark = Data1.Recordset.LastModified
Beep
VideosViewChangeForm.cmdNew.Enabled = True
VideosViewChangeForm.cmdDelete.Enabled = True
End If
Exit Sub
ErrorHandler:
MsgBox "Duplicate Record. No new record created."
VideosViewChangeForm.cmdNew.Enabled = True
VideosViewChangeForm.cmdDelete.Enabled = True
Data1.Recordset.MoveFirst
End Sub
```

```
Private Sub cmdClose_Click()
VideosViewChangeForm.cmdUpdate.Enabled = True
MaintenanceForm.Visible = True
Unload Me
End Sub
```

```
Private Sub Command1_Click()
Unload Me
MaintenanceForm.Visible = True
End Sub
```

```
Private Sub Command2_Click()
Dim Response As Integer
'this may produce an error if you delete the last
'record or the only record in the recordset
Response = MsgBox("Deleting this record will remove it for good!" & Chr(13) & Chr(10) & _
"Are you sure you want to do this?", vbYesNo)
If Response = vbYes Then
If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then
Data1.Recordset.Delete
Data1.Recordset.MoveNext
If Data1.Recordset.EOF = True Then
Data1.Recordset.MovePrevious
End If
Else
MsgBox "There are no records to delete."
End If
End If
```

End Sub

Private Sub Command3_Click()

 Data1.Recordset.AddNew

End Sub

Private Sub Command4_Click()

 If Left(VideosViewChangeForm.txtFields(0), 1) <> "" And _

 Left(VideosViewChangeForm.txtFields(0), 1) <> " " And _

 Left(VideosViewChangeForm.txtFields(1), 1) <> "" And _

 Left(VideosViewChangeForm.txtFields(1), 1) <> " " And _

 Left(VideosViewChangeForm.txtFields(2), 1) <> "" And _

 Left(VideosViewChangeForm.txtFields(2), 1) <> " " And _

 Left(VideosViewChangeForm.txtFields(3), 1) <> "" And _

 Left(VideosViewChangeForm.txtFields(3), 1) <> " " Then

 Data1.UpdateRecord

 If (Data1.Recordset.BOF = False) And (Data1.Recordset.EOF = False) Then

 Data1.Recordset.Bookmark = Data1.Recordset.LastModified

 Beep

 End If

End If

End Sub

Private Sub Data1_Reposition()

 VideosViewChangeForm.cmdUpdate.Enabled = False

 VideosViewChangeForm.cmdNew.Enabled = True

 On Error Resume Next

 Data1.Caption = "Record: " & (Data1.Recordset.AbsolutePosition + 1)

End Sub

Private Sub Form_Activate()

Dim i As Integer

 VideosViewChangeForm.cmdNew.Enabled = True

 VideosViewChangeForm.cmdUpdate.Enabled = False

 If Data1.Recordset.RecordCount = 0 Then

 For i = 0 To 3

 txtFields(i).Enabled = False

 Next

 End If

End Sub

Private Sub txtFields_Change(Index As Integer)

 VideosViewChangeForm.cmdUpdate.Enabled = True

 VideosViewChangeForm.cmdNew.Enabled = False

End Sub

**** VIDEOSTAKEN.FRM ****

THIS SOURCE CODE DEFINES THE VIDEOS TAKEN FORM AND THE OBJECTS INCLUDED ON THE FORM. THE FORM IS USED TO ACCESS THE VIDEOS LISTED ON THE FORM. THIS PORTION OF THE PROGRAM WAS NOT IMPLEMENTED DUE TO NO VIDEOS AVAILABLE. WINDOWS CLOCK.AVI VIDEO WAS USED FOR TESTING PURPOSES ONLY.

Begin VB.Form VideosTakenForm

```
Attribute VB_Name = "VideosTakenForm"  
Attribute VB_Creatable = False  
Attribute VB_Exposed = False  
Option Explicit
```

```
Private Sub Command1_Click()  
    Unload Me  
    ProgenyMainForm.Visible = True  
End Sub
```

```
Private Sub Label2_Click()  
Dim RetVal As String  
    RetVal = Shell("mplay32 d:\winnt\clock.avi", 1)  
    RetVal = Shell("mplay32 c:\vb\progeny\clock.avi", 1)  
End Sub
```

END OF DODDS SOURCE CODE.

LIST OF REFERENCES

1. Bivens, Lester J. and Moore, Patrick W. *The Bottlenose Dolphin: Nature's ATD in SWMCM Autonomous Sonar Platform Technology*, Proceedings of the Autonomous Vehicles in Mine Countermeasures Symposium. Naval Postgraduate School, Monterey, CA., 1995. Pg. 63, 66
2. Anon., Marine Mammal Research Opportunities,
<http://www.nosc.mil/nrad/technology/mammals/3.html>
3. Microsoft Visual Basic. *Programmer's Guide*. Version 4.0. Operating Environment. Microsoft Corporation. 1995. Document No. 58513-TR6. Printed in the United States of America. Pg. 1, 580, 586-587, 589-590
4. Muller, Robert J. *ORACLE: Developer/2000 Handbook*. Osborne McGraw-Hill, Inc., 1996. ISBN 0-07-882180-0, Printed in the United States of America. Pg. 4,55,57,60
5. Freedman, Alan. *The Computer Glossary: The Complete Illustrated Dictionary*. Seventh Edition, American Management Association, 1995. Printed in the United States of America. Pg. 171
6. Berzins, Valdis and Luqi. *Software Engineering with Abstractions*, Addison-Wesley Publishing Company, 1991. Pg 9
7. Abbey, Michael and Corey, Michael J. *ORACLE: A Beginner's Guide*. Osborne McGraw-Hill, Inc., 1995. ISBN 0-07-882122-3, Printed in the United States of America. Pg. 5
8. Microsoft Visual Basic. *Professional Features. Guide to Data Access Objects*, Version 4.0. Professional Edition. Microsoft Corporation. 1995. Document No. DB58499-0795. Printed in the United States of America. Pg. 27-28, 35, 37, 55, 56, 68, 70, 98, 182, 183, 194
9. Microsoft, Visual Basic. *Crystal Reports for Visual Basic User's Manual*. Version 4.0. Operating Environment. Microsoft Corporation. 1995. Document No. DD58500-0795. Printed in the United States of America., pg. 9, 10, 31
10. Microsoft Visual Basic. *Professional Features: Custom Control Reference*, Version 4.0. Professional Edition. Microsoft Corporation. 1995. Document No. DB58499-0795. Printed in the United States of America. Pg. 87, 89, 91, 93
11. Microsoft Visual Basic. *Language Reference*. Version 4.0. Microsoft Corporation. 1995. Document Number DB58514-0795. Printed in the United States of America. Pg. 295-297, 299, 308
12. Fichtelius, Karl-Erik and Sjolander, Sverre. *Smarter than Man?: Intelligence in Whales, Dolphins and Humans*. First American Edition. Pantheon Books, New York, 1972. Printed in the United States of America. Pg. 30

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center2
 8725 John J. Kingman Rd., Ste 0944
 Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library.....2
 Naval Postgraduate School
 411 Dyer Rd.
 Monterey, CA 93943-5101

3. Prof. Robert McGhee1
 Department of Computer Science, Code CS/Mz
 Naval Postgraduate School
 Monterey, CA 93943

4. Patrick W. Moore.....1
 NCCOSC RDTE DIV D351
 49650 Acoustic Rd., Rm. 108
 San Diego, CA 92152-6254

5. Randy Brill.....1
 NCCOSC RDTE DIV D351
 49650 Acoustic Rd., Rm. 108
 San Diego, CA 92152-6254

6. Michael Dodds.....4
 186 Vance St.
 Chula Vista, CA 91910

7. Galen and Sherri Dodds.....1
 3560 S. Greythorne Way
 Chandler, AZ 85248